VŠB – Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Inteligentní dům s využitím mikrokontrolérů

Intelligent House with using Microcontrollers

Student: Bc. Jiří Beseda
Osobní číslo: BES0012
Vedoucí diplomové práce: doc. Ing. Jaromír Škuta, Ph.D.

Ostrava 2020

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání diplomové práce

Student: **Bc. Jiří Beseda**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 3902T004 Automatické řízení a inženýrská informatika

Téma: Inteligentní dům s využitím mikrokontrolérů
Intelligent House with using Microcontrollers

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Vypracujte přehled vybraných komerčních řešení inteligentních domů. Zaměřte se na komunikační rozhraní, funkčnost modulů a distribuci funkcí.
2. Seznamte se s moduly Arduino a rozšiřujícími moduly, proveďte jejich porovnání a rozdělení s ohledem na jejich technické parametry a komunikační možnosti.
3. Navrhněte řešení pro inteligentní dům s využitím vybraných modulů. Proveďte návrh systémového řešení, vazeb, definujte funkce a způsob ovládání inteligentního domu a toto řešení realizujte.
4. Zhodnoťte dosažené výsledky a navrhněte směr dalšího řešení.

Seznam doporučené odborné literatury:

HRBÁČEK, Jiří. *Moderní učebnice programování jednočipových mikrokontrolérů PIC* [CD-ROM]. 1. díl, První krůčky při tvorbě aplikace. Praha: BEN - technická literatura, 2004. ISBN 80-7300-136-5.
MARGOLIS, Michael. *Arduino cookbook.* 2nd ed. Sebastopol: O'Reilly, c2012. ISBN 978-1-449-31387-6.
VALEŠ, Miroslav. *Inteligentní dům.* Brno: ERA, 2006. ISBN 80-7366-062-8.
VODA, Zbyšek. *Průvodce světem Arduina.* Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jaromír Škuta, Ph.D.**

Datum zadání: 20.12.2019

Datum odevzdání: 18.05.2020

VYSOKÁ ŠKOLA BÁŇSKÁ TECHNICKÁ UNIVERZITA Fakulta strojní OSTRAVA

doc. Ing. Renata Wagnerová, Ph.D.
vedoucí katedry

prof. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě dne 18. května 2020.

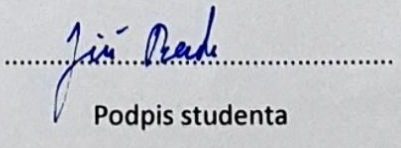
.....

Podpis studenta

Prohlašuji, že:

- jsem si vědom, že na tuto moji závěrečnou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. Zákon o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (dále jen Autorský zákon), zejména § 35 (Užití díla v rámci občanských či náboženských obřadů nebo v rámci úředních akcí pořádaných orgány veřejné správy, v rámci školních představení a užití díla školního) a § 60 (Školní dílo),
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO") má právo užít tuto závěrečnou diplomovou práci nekomerčně ke své vnitřní potřebě (§ 35 odst. 3 Autorského zákona),
- bude-li požadováno, jeden výtisk této diplomové práce bude uložen u vedoucího práce,
- s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 Autorského zákona,
- užít toto své dílo, nebo poskytnout licenci k jejímu využití, mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše),
- beru na vědomí, že podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů - že tato diplomová práce bude před obhajobou zveřejněna na pracovišti vedoucího práce a v elektronické podobě uložena a po obhajobě zveřejněna v Ústřední knihovně VŠB-TUO, a to bez ohledu na výsledek její obhajoby.

V Ostravě dne 18. května 2020.



Podpis studenta

Anotace

BESEDA, J. Inteligentní dům s využitím mikrokontrolérů: Diplomová práce . Ostrava: VŠB - technická univerzita Ostrava, Fakulta strojní, Katedra automatizační techniky a řízení, 2020, Vedoucí práce: Škuta, J.

Tématem diplomové práce je inteligentní domácnost s využitím mikrokontrolérů. Jako mikrokontrolér vhodný pro inteligentní domácnost je platforma Arduino. Před návrhem samotného řešení inteligentní domácnosti práce zmiňuje komerční řešení a způsoby jejich komunikace. Zaměřuje se zejména na řešení Philips Hue a standard KNX, které jsou také v druhé polovině práce napojený na řešení využívající Arduino. Je navrženo a popsáno několik modulů řešení Arduino a poté představeno jejich zapojení jako celku v jednom řešení.

Klíčová slova: Arduino, Smart Home, Domácí automatizace, KNX, Philips Hue

Anotation

Beseda, J. Smart home with using microcontrolers: Diploma Thesis. Ostrava: VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 2020, Thesis head: Škuta, J.

This diploma thesis is about intelligent house with using of Microcontrollers. As suitable choice for this task is selected platform Arduino. Before designing solution for intelligent house thesis describes commercial solutions and their communications, focusing on solution Philips Hue and standard KNX, which is in second half of thesis connected to solution using Arduino. Thesis also describes several modules which are used in Arduino solution and then whole solution is described.

Key words: Arduino, Smart Home, Home automation, KNX, Philips Hue

Seznam použitých symbolů a zkratek

3D	Trojrozměrné (*Three dimensions*)
AC	Střídavý proud (*Alternating current*)
API	Rozhraní pro programování aplikací (*Application Programming Interface*)
DC	Stejnosměrný proud (Direct Current)
EIB	Evropská instalační sběrnice (*European Instalation Bus*)
GPRS	General Packet Radio Service
I/O	Vstup/Výstup (Input / output)
IDE	Vývojové prostředí (Integrated Development Environment)
IP	Internetový protokol (*Internet Protocol*)
IR	Infračervený (*Infra Red*)
JSON	JavaScriptový Objektový Zápis (*JavaScript Object Notation*)
KNX	Mezinarodní standard zařízení
LAN	Lokální síť (*Local Area Network*)
LED	Světlo emitující dioda (*Light Emitting Diode*)
MCU	Jednočipový počítač (*Micro Controler Unit*)
MOSFET	Typ tranzistoru (*Metal Oxide Semiconductor Field Effect Transistor*)
PAN	Osobní síť (*Personal Area Network*)
PID	Proporčně Integračně Derivační
PL	Silové vedení (*Power Line*)
PWM	Pulzně šířková modulace (*Pulse Width Modulation*)
RF	Radiová frekvence (*Radio Frequency)*
RS485	Standard sériové komunikace
SMS	Služba krátkých textových zpráv *(Short Message Service)*
TP	Kroucený pár (*Twisted Pair*)
TTL	Tranzistorově Tranzistorová Logika (*Transistor-Transistor Logic*)
URL	Jednotná adresa zdroje (*Uniform Resource Locator*)
USB	Univerzální sériová sběrnice (*Universal Serial Bus*)
Wi-Fi	Bezdratové připojení (*Wireless Fidelity*)

Obsah

Úvod

Před desítkami let lidé v domácnostech svítili svíčkami, petrolejovými lampami a topili dřevem nebo uhlím. Toto bylo nebezpečné a často vznikaly požáry. Avšak v průběhu 19. století byla objevena elektřina a nedlouho poté Thomas Edison vynalezl žárovku, Graham Bell vynalezl telefon a mnoho dalších vynálezců vytvořilo zařízení, které využíváme dodnes, i když v jiných podobách. Elektřina změnila celou společnost, ulehčila práci, zkrátila vzdálenosti ve světě a stala se neodmyslitelnou součástí lidského života.

Dalším velkým skokem byl rok 1947, kdy došlo k objevu tranzistorového jevu. Tranzistory a tranzistorová logika jsou základními kameneny všech běžných elektronických přístrojů, počítačů, mobilních telefonů, moderních praček a ledniček. Od vynálezu tranzistoru došlo k jejich výrazné miniaturizaci a v současnosti mají velikosti v jednotkách nanometrů. Moderní přístroje usnadňují jejich vlastníkům život, šetří velké množství času a vzdálenosti ve světě mezi lidmi se díky nim staly zanedbatelnými.

Tranzistory jsou základem počítačů a počítače se používají k automatizaci práce bez dohledu člověka. Automatizace průmyslu umožnuje strojům pracovat nepřetržitě a s přesností, které by člověk nikdy nemohl dosáhnout. Avšak tyto počítače a stroje jsou velmi drahé kvůli požadavkům na přesnost, stabilitu výkonu a náročnost uvedení do provozu. Proto taková řešení nejsou vhodná do domácností a tento stav přetrvával po dlouhou dobu.

Dnes se cena počítačů snížila a lze vytvořit malé přístroje za velmi nízké ceny. Díky tomuto se může domácnost opět posunout o krok dále. A skok je srovnatelný s přechodem od petrolejové lampy k žárovce. Jednotlivé prvky domácnosti lze propojit a vytvořit systém, který se stará o všechny potřeby domu a jeho obyvatel.

Představme si modelový den s chytrým domem. Při východu slunce se automaticky roztáhnou závěsy a nastaví žaluzie. Toto umožní maximální využití tepla ze slunečního záření v zimě. Po zmáčknutí tlačítka vedle postele se začnou nahřívat ručníky v koupelně na otopném tělese a v kuchyni kávovar začne připravovat kávu. Při sprchování dům rozpozná zvýšenou vlhkost v koupelně, a proto zvýší intenzitu větrání v domě a zapne systém pro odmlžení zrcadla. Po připravení a odchodu do práce se automaticky sníží intenzita větrání v domě a zhasnou všechna světla. V průběhu dne zazvoní pošťák a za použití mobilního telefonu je možné doručovatele s balíčkem pustit dovnitř. Když se výrazně zvýší počet osob v domě, senzory rozpoznají vyšší hladinu oxidu uhličitého a zvýší intenzitu větrání. Při

setmění se zatáhnou žaluzie a dům plynule uzpůsobí intenzitu světla podle venkovních senzorů. Před ulehnutím stačí zmáčknout tlačítko a dům se připraví na noc, zatáhne žaluzie, zhasne světla, sníží teplotu, na kterou se v zimě vytápí, zapne zalévání zahrady a zapne vnější alarm.

Z modelového příkladu je zřejmé, že takový dům přináší výrazné zvýšení komfortu. Technologie pro takový dům jsou již dostupné delší dobu. V posledních letech však cena těchto technologií klesá a vznikají platformy, díky kterým si člověk může vytvořit vlastní řešení pro chytrou domácnost. Avšak stále platí, že kvalitní řešení jsou drahá a pro jejich realizaci je potřeba oslovit profesionální firmu.

1 Komerční řešení pro chytrou domácnost

V dnešní době masivní automatizace průmyslu dochází ke snižování cen zařízení pro automatizaci a tyto technologie se tedy stávají dostupné i pro využití v domácnostech. Systémy obsahující tato zařízení obsahují moduly vstupů jako například senzory a spínače a různé výstupní prvky umožňující spínání zařízení, například světla či motory.

Obr. 1-1 Chytrý dům (1)

Dům, který je znázorněn na obr. 1-1. obsahuje velké množství senzorů a je schopen automaticky reagovat na jimi naměřené hodnoty. Možnosti využití závisí na použitém systému. Základní systémy, které si každý může nainstalovat v domácnosti sám, zvládají většinou jen základní úkony jako je ovládání světel a zásuvek. Profesionální sběrnicové systémy zvládnou pokročilejší úkony, například automaticky podle úrovně svitu ovládat venkovní žaluzie, sledovat kvalitu vzduchu uvnitř domu a v případě vysoké úrovně vlhkosti nebo CO_2 zvýšit intenzitu větrání. Nejpokročilejší systémy poté zvládají i PID regulaci.

V této kapitole bude popsáno několik typů řešení:

- sběrnicový standard KNX, který patří k profesionálním řešením;
- Philips Hue, základní řešení zaměřené na ovládání světel;
- Další méně rozšířená řešení.

1.1 KNX standard

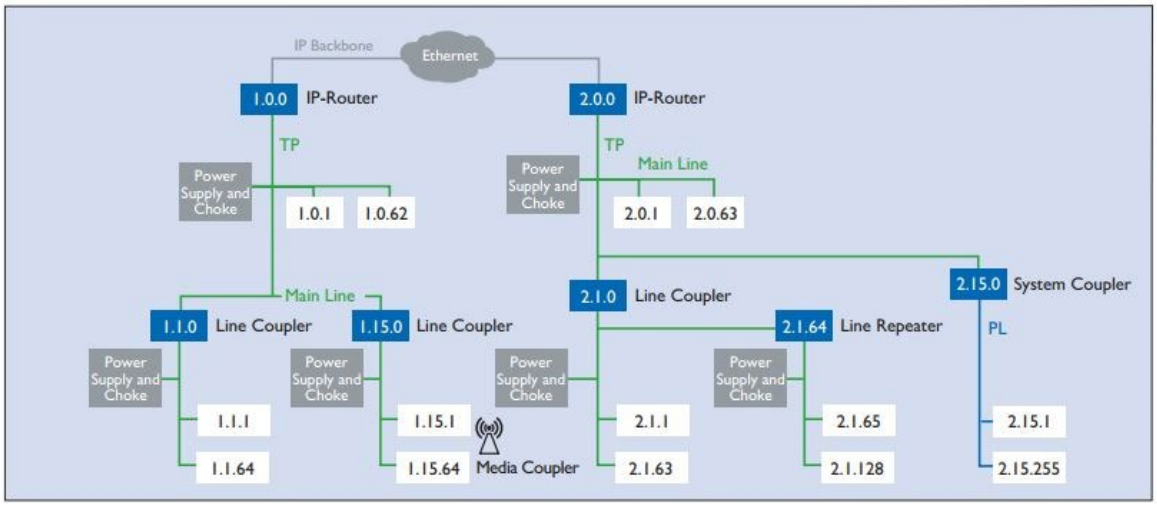
KNX standard je mezinárodní standard sběrnicového řešení, podle kterého výrobci několika značek vyrábí zařízení, která navzájem komunikují a umožňují ovládat všechny prvky chytré domácnosti, od světel až po automatické ovládání větrání. Nevýhodou tohoto řešení je, že vyžaduje odbornou instalaci a v ideální případě zcela jiný způsob elektroinstalace, než je běžné zapojení. Proto není toto řešení vhodné pro instalaci domácími kutily.

1.1.1 Architektura řešení KNX

Zprávy, které jsou posílány po sběrnici KNX jsou dvojího typu:

- Zprávy s adresátem - je přesně určeno, pro jaké zařízení je zpráva určena a toto zařízení zasílá potvrzující informaci o přijetí zprávy.
- Obecné zprávy - například naměřená hodnota senzoru či čas. Zprávy tohoto typu jsou určeny pro všechna zařízení na sběrnici a po jejich přijetí se neposílá žádná zpětná vazba. Zprávy posílané po sběrnici nejsou šifrovány, a je tedy třeba zajistit zabezpečit sběrnici proti vnějším útokům.

Vzhledem k tomu, že ve větších instalacích je potřeba zajistit dostatečnou propustnost sběrnice a zároveň z důvodu bezpečnosti a přehlednosti oddělit části systému, má KNX hierarchickou topologii (obr. 1-2). Adresa se skládá ze tří číslic oddělených tečkami. První číslo označuje oblast, druhé linii a třetí adresu samotného zařízení. Tato topologie umožňuje umístit do jedné instalace až 57375 zařízení. (2)



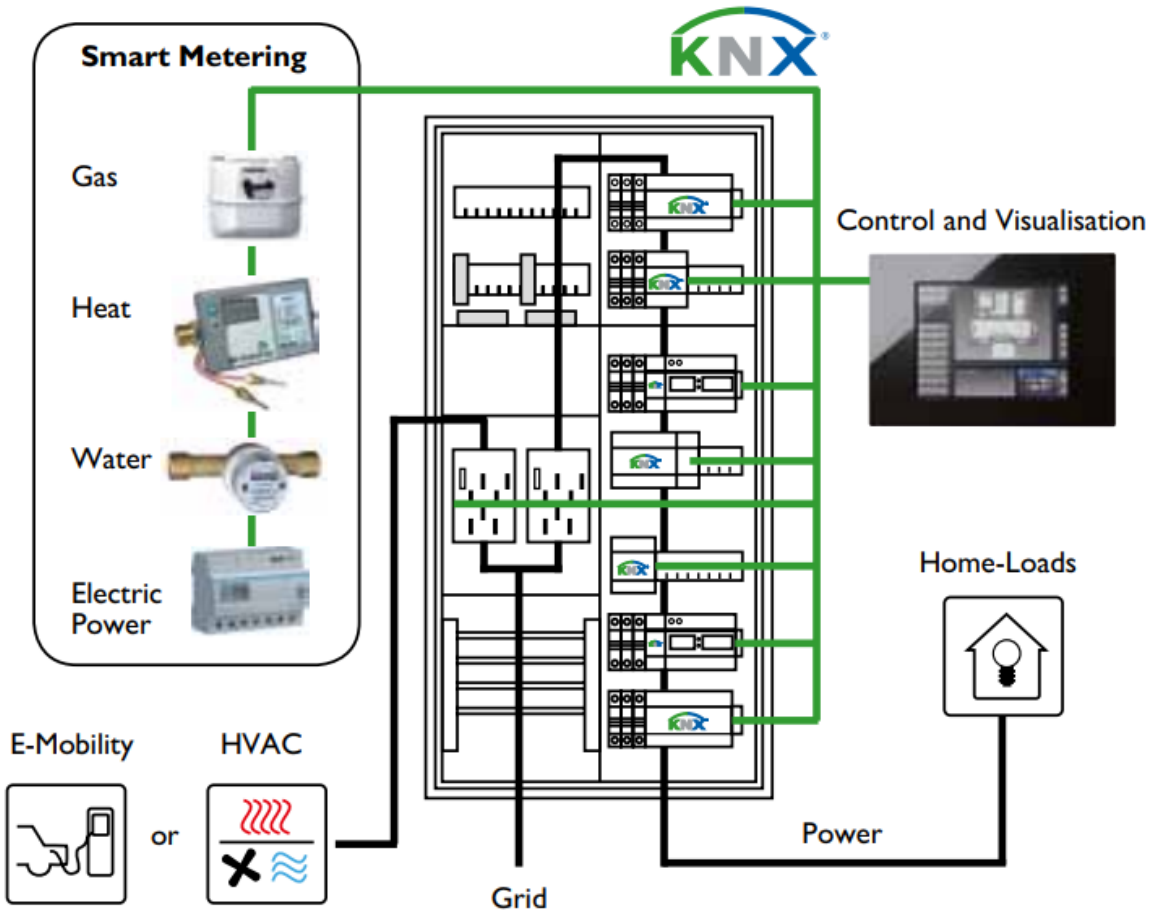
Obr. 1-2 Topologie KNX (2)

1.1.2 Komunikace na sběrnici KNX

V současnosti existují dle stránek asociace KNX čtyři druhy přenosových medií pro sběrnici (3)

1) Kroucený pár (TP)
   – Nejpoužívanější řešení přejaté od standardu EIB;
   – Dva vodiče, ve kterých se nachází napájení i samotný signál sběrnice;
   – Bitrate je 9600 bit/s;
   – Většinou je využíván vodič typu J-Y(ST)Y 2x2x0,8. V tomto kabelu jsou celkem 4 vodiče, červený a černý je běžně používán pro KNX, žlutý a bílý slouží jako záložní nebo je využíván jako dodatečné napájení zařízení.
2) Power line (PL)
   - Přejaté řešení od standardu EIB;
   - Komunikace využívající elektroinstalační síť v budově;
   - Bitrate 1200 bit/s;
   - Lze aplikovat v budovách, kde není možné provést instalaci krouceného páru.
3) Radiová frekvence (RF)
   - Komunikace probíhá prostřednictvím radiových vln na frekvenci 868 MHz s výkonem 25mW;
   - Bitrate 16,384 Kbit/s;
   - Není nutné instalovat žádné kabely, lze použít v nových budouvách i při rekonstrukcích.
4) Ethernet (IP)
   - Využita je ethernetová síť;
   - Nejrychlejší komunikace;
   - Zprávy jsou posílány uvnitř domácí sítě nebo internetem.

Nejvyužívanější formou komunikace je kroucený pár pro větší mechanickou odolnost a umožnění přenášet data vetší rychlostí. Toto řešení je spolehlivé, bezpečné a zároveň umožňuje napájet zařízení z jednoho zdroje. Schéma takového zapojení lze vidět na obr. 1-3, na kterém je i znázorněno napojení KNX sběrnice na domácí síť v rámci rozvodné skříně.



Obr. 1-3 Architektura KNX (2)

1.1.3 Zařízení KNX

V současné době jsou výrobky využívající standard KNX součástí produktů mnoha firem. Jsou to malé společnosti jako MDT, středně velké společnosti jako JUNG či GIRA nebo velké společnosti jako ABB či Schneider Electrics. Díky tomu, že se jedná o mezinárodní standard, tak existují zařízení pro všechny aplikace, které by vyžadoval uživatel pro automatizaci své domácnosti. Nevýhodou těchto zařízení je vysoká cena, která se pohybuje většinou mezi 10 až 30 tisíci Kč za zařízení, v extrémních případech je cena i vyšší než 100 tisíc Kč. Ukázky zařízení jsou na obr. 1-5.

Obr. 1-5 Zařízení KNX (16)

Server Gira X1

Většina výrobců využívajících standard KNX má ve své nabídce zařízení, které lze označit jako server celé domácnosti. Výjimkou není ani firma Gira, která nabízí zařízení X1 (obr. 1-6). Toto zařízení nabízí mnoho funkcí, hostuje aplikaci pro ovládání KNX systému, umožňuje napojení KNX sběrnice a ethernetu, lze v něm nastavit časovače, logické prvky a mnoho dalšího.

Obr. 1-6 Gira X1 (4)

Gira X1 se nastavuje v programu Gira Project Asistant. GPA slouží k nastavení časovačů, logických prvků, správě uživatelů a zejména k samotnému nastavení zařízení a rozhraní pro

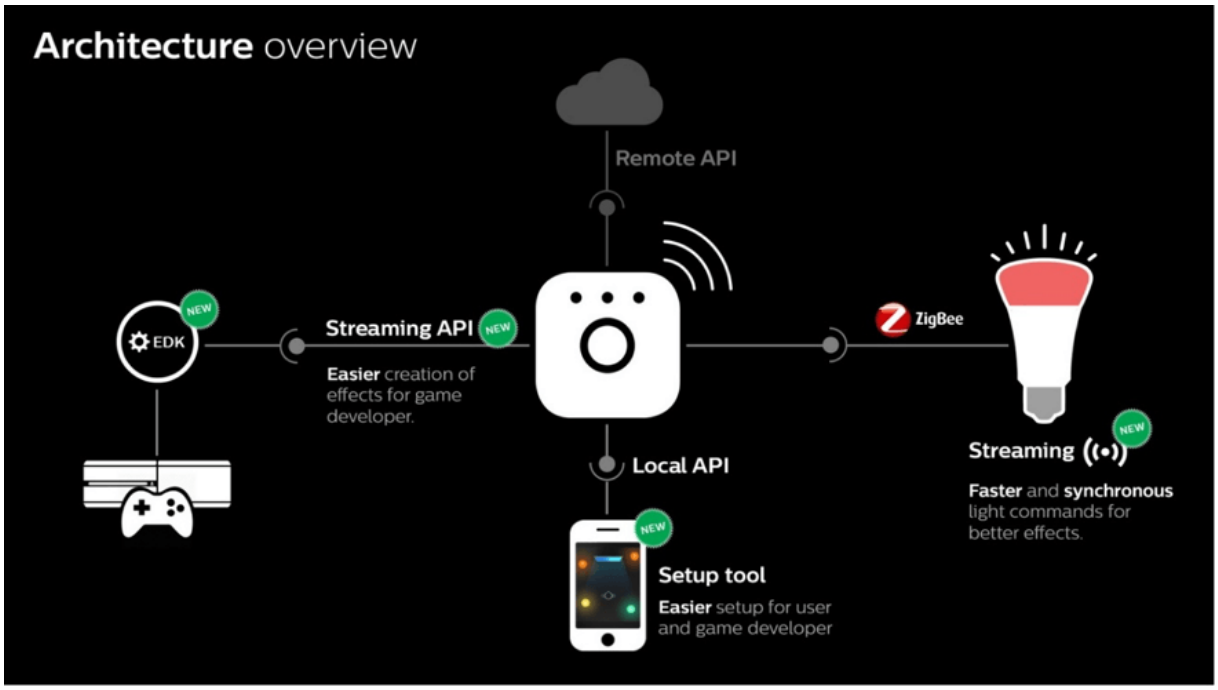
ovládání pomocí mobilních aplikací. Mobilní rozhraní má mnoho prvků, které lze použít, např. vypínače, zobrazování hodnot, ovládání teploty, zobrazení záběru kamery nebo URL call. URL call lze využít pro napojení na jiné systémy domácí automatizace a jejich ovládání. Gira X1 lze použít jako centrální server pro inteligentní dům kombinující více typu řešení.

1.2 Philips Hue

Jedná se o řešení od firmy Philips, které se zaměřuje na osvětlení. Philips nabízí různé typy světel, žárovek a LED pásků. Výhodou je, že všechna zařízení jsou navržena tak, že je dokáže spravovat i průměrný uživatel bez znalosti programování. Velkou nevýhodou je, že standardně nelze ovládat domácnost mimo domácí síť a také, že možnosti jsou omezené pouze na ovládání světel. Proto je vhodné toto řešení používat v kombinaci s jinými řešeními inteligentních domácností.

1.2.1 Architektura řešení - Zigbee

Philips Hue využívá zařízení s názvem „Bridge“, které je napojené na domácí síť a komunikuje s ostatními prvky řešení. Zpracovává příchozí zprávy z LAN sítě a přeposílá je ostatním zařízením přes ZigBee protokol. Toto umožňuje vytvořit komunikaci, kterou lze vidět na obr. 1-7.



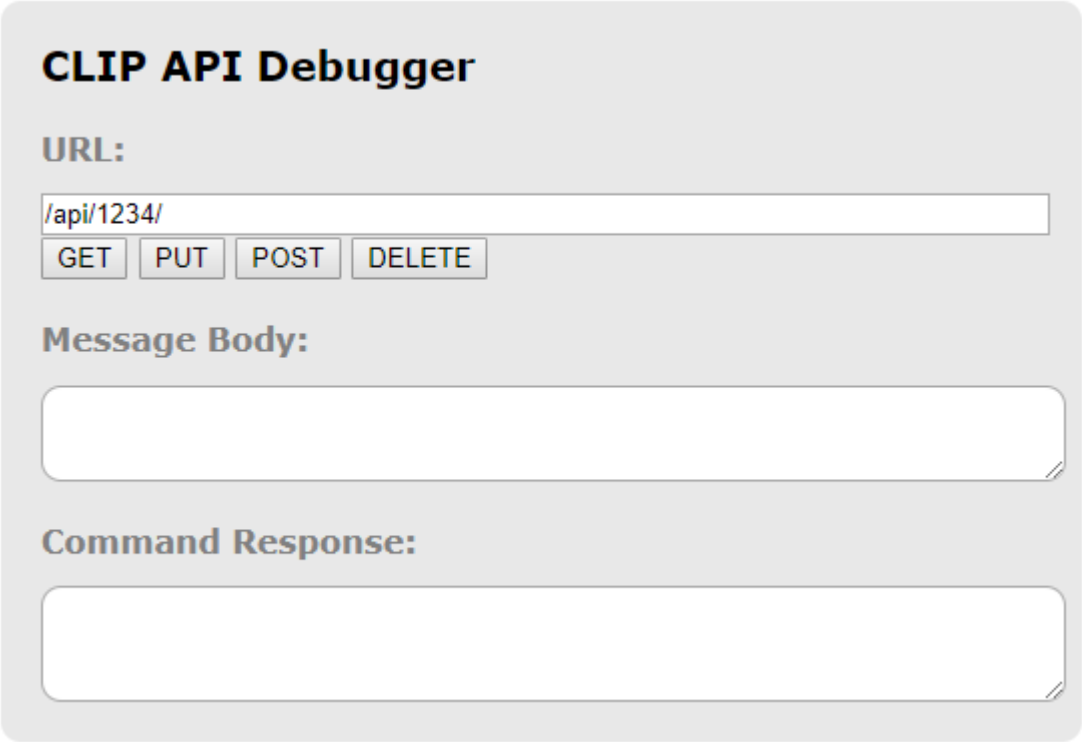
Obr. 1-7 Architektura řešení Philips Hue (5)

Jak je zřejmé z popisu výše, celé řešení je postaveno na komunikaci pomocí protokolu ZigBee. Tento protokol je označován jako IEEE 802.15.4. Patří mezi PAN (Personal Area Networks) stejně jako Bluetooth či Wi-Fi a komunikace probíhá na frekvenci 2,4 GHz.

ZigBee slouží pro komunikaci, u které je vyžadováno, aby byla nenáročná na energii, a proto s sebou přináší jistá omezení. Komunikační rychlost je definována na rychlost 250kbit/s a maximální vzdálenost 100 metrů, obvykle však pouze 10 - 30 metrů. Jedná se o protokol vhodný pro malé sítě a přenosy malých datových packetů jako hodnoty senzorů či krátké příkazy jako rozsvítit/zhasnout světlo.

1.2.2 Komunikační API Philips Hue

Komunikace pomocí Bridge funguje na principu RESTful interface a všechny příkazy jsou dobře popsány na stránkách Philips Hue pro vývojáře. Pokud se nacházíme na stejné LAN jako Philips Hue Bridge, tak můžeme také využít webové rozhraní (obr. 1-8) pro posílaní API požadavků na adrese.



Obr. 1-8 Philips Hue web debugger

Tento Debugger má všechny funkce potřebné ke správnému zavolání HTTPS požadavku na Hue RESTful interface.

1) URL – Adresa zařízení/objektu, v Hue řešení. Může se jednat o světlo, skupinu světel či další prvky Hue řešení. Požadovaná akce poté bude vykonána na tomto objektu.
2) Metoda – Existují 4 metody, které lze v Hue požadavku použít:
    a. GET – Získá všechny informace o objektu určeném v URL;
    b. PUT – Příkaz pro změnu adresovaného objektu v URL;

c. POST – Příkaz pro vytvoření nového objektu v adresovaném objektu;
d. DELETE – Příkaz pro vymazání adresovaného objektu.

3) Message Body – Zde se vkládá JSON, který upřesňuje informace pro požadovanou změnu.
4) Command Response – Zde se po odeslání objeví odpověď na požadovaný příkaz.

1.2.2.1 Vytvoření uživatele

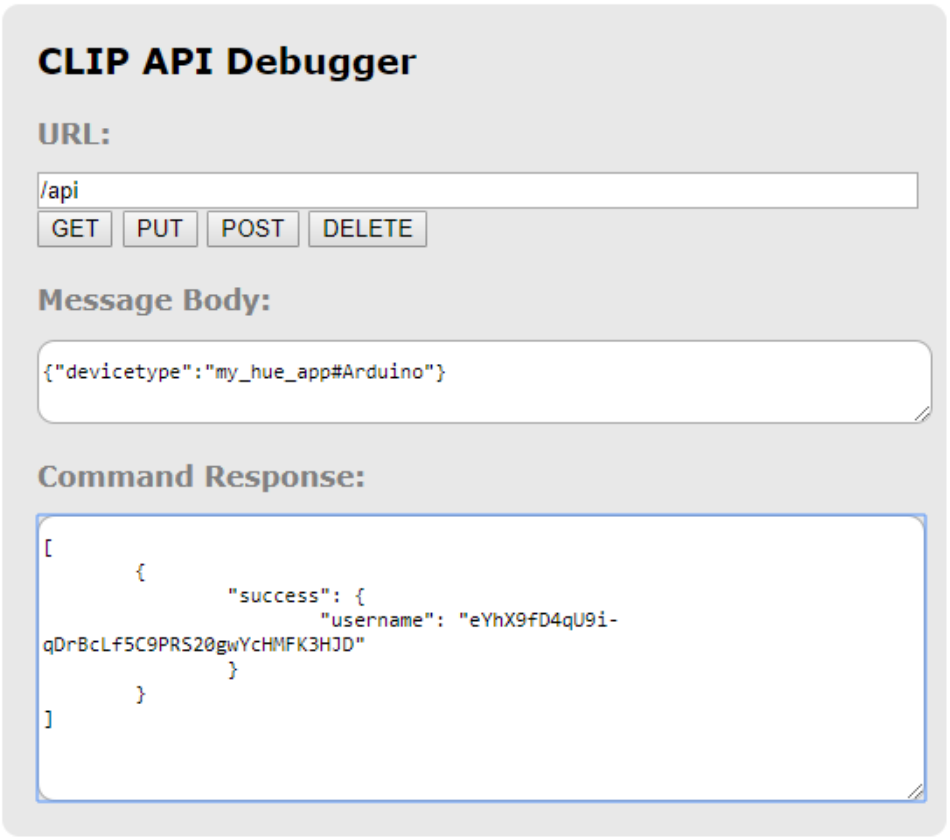
Prvním krokem pro použití API debugger je vytvoření uživatele. Toho docílíme zadáním následujících parametrů:

URL - /api

Metoda – POST

Message Body: {"devicetype":"my_hue_app#Arduino"}

Důležité je, že před odesláním samotného příkazu je potřeba zmáčknout tlačítko na Philips Hue Bridge pro ověření, že uživatel má přístup k zařízení. Po odeslání získáme username (obr. 1-9), které je nutné pro ovládání Philips Hue. V našem případě je username „eYhX9fD4qU9i-qDrBcLf5C9PRS20gwYcHMFK3HJD“ a v dalším vývoji budeme vždy používat toto username.



Obr. 1-9 Vytvoření nového uživatele

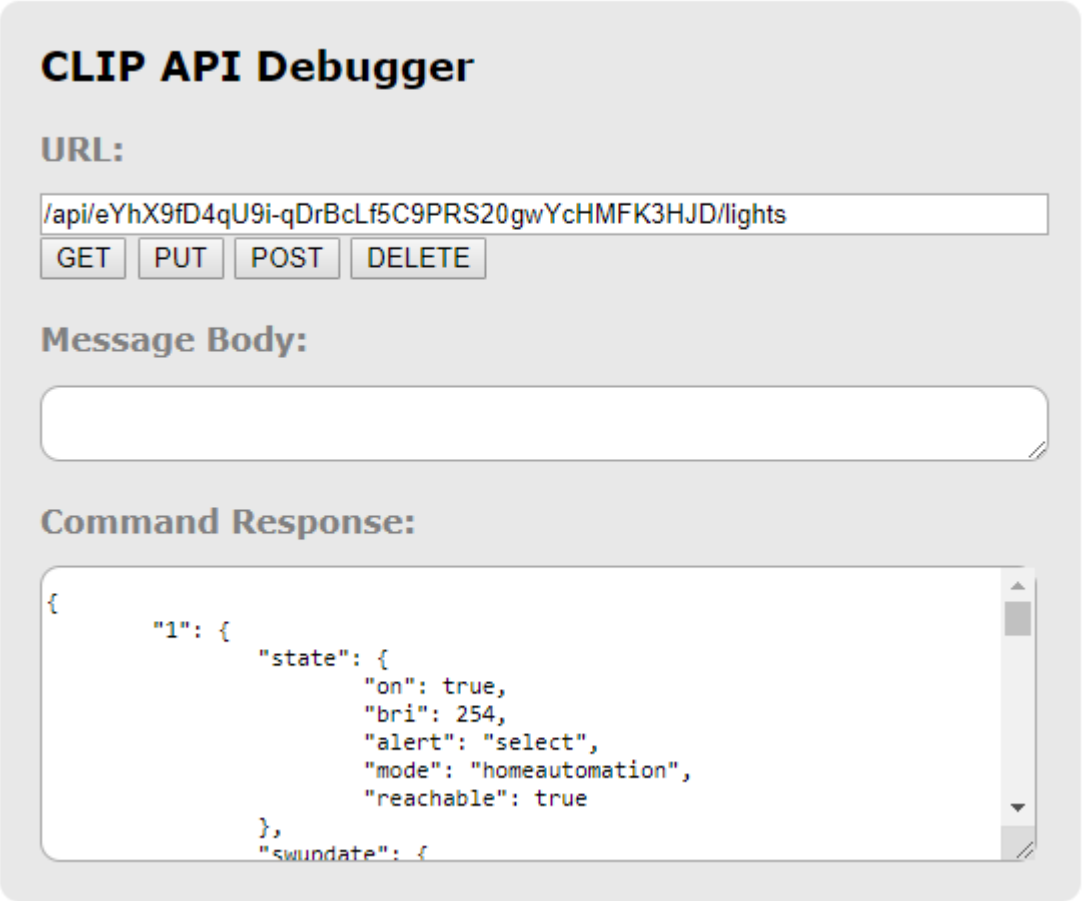
1.2.2.2 Světla

Nyní máme vytvořeného uživatele a lze tedy ovládat jednotlivé objekty v Hue řešení. Další příkaz, který použijeme, je získání informace o tom, jaká světla se nachází v systému.

URL - /api/eYhX9fD4qU9i-qDrBcLf5C9PRS20gwYcHMFK3HJD/lights

Metoda – GET

Odpovědí systému je JSON obsahující informace o všech světlech zapojených v řešení, jak lze vidět na obrázku. Z obrázku je patrné, že existuje světlo s ID 1 a můžeme ho tedy v dalším kroku ovládat (obr. 1-10).



Obr. 1-10 Informace o světlech v řešení

1.2.2.3 Ovládání světel

Nejobvyklejším příkazem Philips Hue je rozsvícení/zhasnutí světel či nastavení jasu. Pro rozsvícení/zhasnutí světel se použije příkaz:

URL - /api/eYhX9fD4qU9i-qDrBcLf5C9PRS20gwYcHMFK3HJD/lights/1/state

Metoda – PUT

Message Body - {"on":false} pro zhasnutí, {"on":true} pro rozsvícení

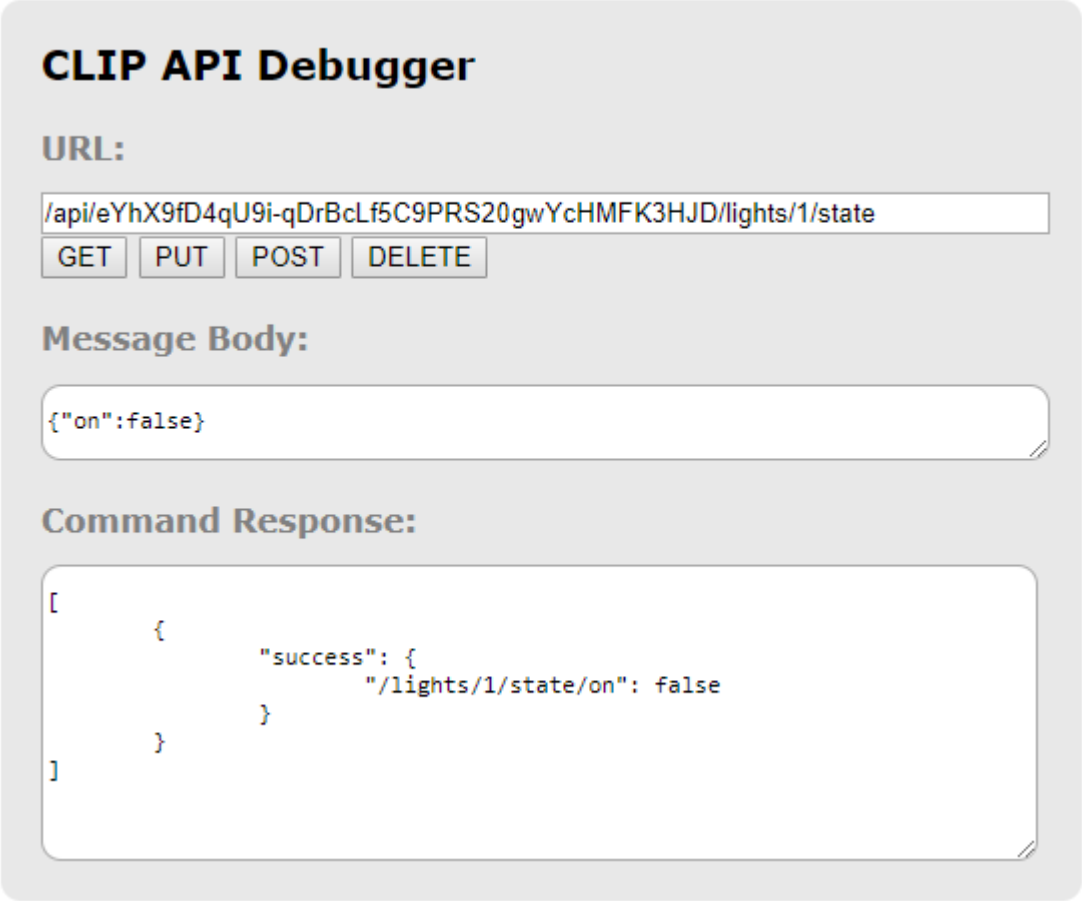
Pro nastavení jasu světla je poté využit příkaz:

URL - /api/eYhX9fD4qU9i-qDrBcLf5C9PRS20gwYcHMFK3HJD/lights/1/state

Metoda – PUT

Message Body - {"on":true, „bri“:128 }

Odpověd na zhasnutí světla lze vidět na obr. 1-11, kdy došlo k úspěšnému zhasnutí světla, v opačném případě by odpovědí byla chybová hláška.



Obr. 1-11 Odpověď na úspěšně zhasnutí světla

Existuje velké množství příkazů, které lze použít pro ovládání nebo získání informací, všechny lze najít v dokumentaci Philips Hue, která je volně dostupná na internetu. Philips uvádí, že veškerá poškození zapříčiněná používáním těchto příkazů jsou na vlastní riziko a Philips za ně nenese zodpovědnost.

1.3 Další komerční řešení

Vzhledem k velkému zájmu o chytrou domácnost existuje velké množství dalších řešení, ve většině případů se ovšem nejedná o řešení dosahující kvalit dvou dříve zmíněných systémů. Často se jedná o řešení, která jsou vyvíjena jako open source řešení nadšenci do domácí automatizace.

1.3.1 Centralizovaná řešení a asistenti

Jeden z typů řešení jsou zařízení, která slouží jako centrální ovládací jednotka. Tato zařízení nemají žádné přímé vstupy či výstupy. Účel těchto centralizovaných zařízení je napojit se na všechna dostupná řešení v domácnosti a umožnit uživatelům jeden přístupový bod pro ovládání. Komunikační rozhraní pro propojení je domácí síť či internet.

Home assistant

Pokud už v domácnosti je několik řešení různých výrobců a je zájem tato řešení sjednotit a vytvořit jeden přístupový bod pro všechna řešení, tak tento server je ideální volbou. Home Assistant je open source řešení domácího serveru, které umožňuje ovládání mnoha dostupných řešení domácí automatizace prostřednictvím webového rozhraní. Home Assistant je možné nainstalovat na jakýkoli počítač, zcela dostačující je malý jednodeskový počítač jako je např. Raspberry Pi.

V současnosti podle dokumentace Home assistant je možné napojit server na desítky jiných komerčních řešení. Napojení jsou možná na oblíbené systémy jako Philips Hue, ale třeba také na odjezdy vlaků Deutche Bahn. (6)

IFTT

Oproti předchozímu řešení se jedná o plně virtuální řešení. Název tohoto řešení je zkratka pro „If This Then That“ což je v překladu „Když toto tak tamto“ a tento název přesně vystihuje princip tohoto řešení. Pomocí něj lze propojit různá řešení, které využívají internet a mají API. Takže například ve chvíli, kdy člověk opustí dům, se automaticky vypnou všechna Philips Hue světla v domě. (7)

Amazon Alexa, Google assistant

Obě tato řešení jsou domácí asistenti s umělou inteligencí. Ovládání je hlasové a díky tomu mohou být oba asistentni využiti v zařízeních mnoha výrobců, primárně se ovšem nachází ve výrobcích od svých firem Amazon Echo v případě Alexy a Google Home v případě Google Assistant. Asistenti mají mnoho funkcí od předpovědi počasí přes ovládání hudby po ovládání domácnosti pomocí IFTT, umožňují ovládat i Philips Hue nebo Fibaro a další. (8) (9)

1.3.2 Decentralizovaná řešení chytré domácnosti

Pro plné využití chytré domácnosti je zapotřebí využít jedno z decentralizovaných řešení, které pokrývá všechny potřeby obyvatel domácnosti a domu samotného. Většina řešení spoléhá na bezdrátovou komunikaci využívající jeden z dostupných komunikačních standardů - Z-Wave, ZigBee či jiné.

Loxone

Loxone je profesionální sběrnicové řešení velice podobné standardu KNX. Na rozdíl od něj, se v tomto případě jedná o řešení jedné firmy a implicitně jej nelze připojit na jiná řešení.

Topologie Loxone je stejná jako topologie KNX - stromová struktura se spínacími relé v rozvodné skříni, kde se také nachází centrální jednotka a její rozšíření. Samotná sběrnice je čtyřvodičová, dva vodiče se signálem a dva vodiče s napájením zařízení. Doporučený vodič použitý jako sběrnice je CAT7 kabel obsahující celkem 8 vodičů se stíněním od vnějších elektromagnetických vlivů.

Toto řešení pokrývá většinu standardní potřeb chytrého domu a jedná se o adekvatní alternativu za zařízení standardu KNX. (1)

Fibaro

Fibaro je jedno z bezdrátových řešení, centrální jednotka je napojena na domácí síť a tím umožňuje napojení na internet. Pro komunikaci zařízení v rámci řešení Fibaro je využit standard Z-Wave, který je využíván primárně v domácí automatizaci. Jedná se o bezdrátovou komunikaci na rádiových frekvencích 800-900 MHz s teoretickým dosahem až 100 metrů, praktickým dosahem přibližně 25 metrů při využití v budovách. Použitá frekvence záleží na státě, ve kterém je zařízení použito. Ne všechna zařízení musí být v přímém dosahu a Z-Wave vytváří mesh síť. Každé zařízení tedy rozšiřuje dosah všech ostatních zařízení a je nutné zajistit, aby žádné zařízení nebylo izolováno bez dosahu k jinému zařízení. (10) (11)

Řešení lze porovnávat podle mnoha parametrů. Tabulka 1 porovnává řešení s ohledem na způsob komunikace, zabezpečení a typ řešení.

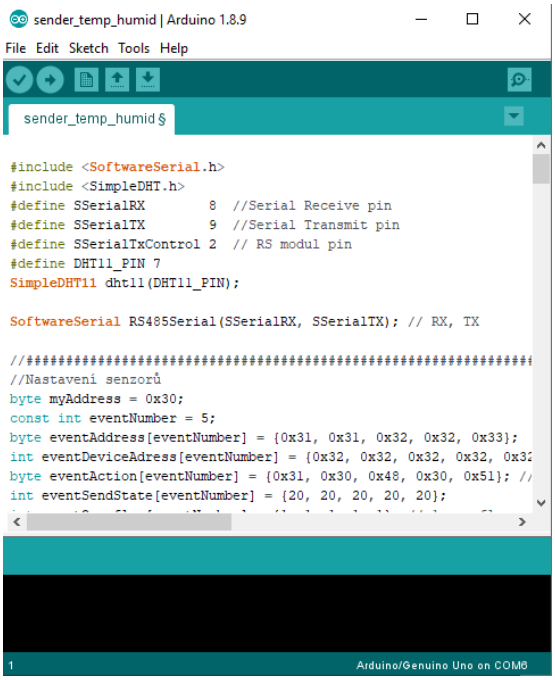
Tabulka 1 Přehled řešení chytré domácnosti

Název řešení	Typ řešení	Způsob komunikace	Zabezpečení
Standard KNX	Decentralizované	Sběrnice (2 vodiče)	Žádné
Philips Hue	Decentralizované	ZigBee	AES 128bit klíč
Home Assistant	Centralizované	Ethernet	SSL/TLS/SSH
IFTT	Centralizované	Ethernet	SSL/TLS
Amazon Alexa	Centralizované	Ethernet	SSL/TLS
Google Assistant	Centralizované	Ethernet	SSL/TLS
Loxone	Decentralizované	Sběrnice (4 vodiče)	Žádné
Fibaro	Decentralizované	Z-Wave	S2 (Dříve AES)

2 Vybrané desky a shieldy Arduino

Informace v této kapitole jsou čerpány z produktových stránek firmy Arduino. (12)

Desky Arduino jsou jednodeskové počítače založené na mikročipu ATmega od firmy Atmel. Podle typu použitého mikročipu poskytuje několik digitálních a analogových I/O pinů. Využívá se jazyka wiring a programuje se v open-source prostředí Arduino IDE (obr. 2-1). Díky tomu je programování velice jednoduché, a proto je vhodné i pro výuku na školách a pro hobby projekty.



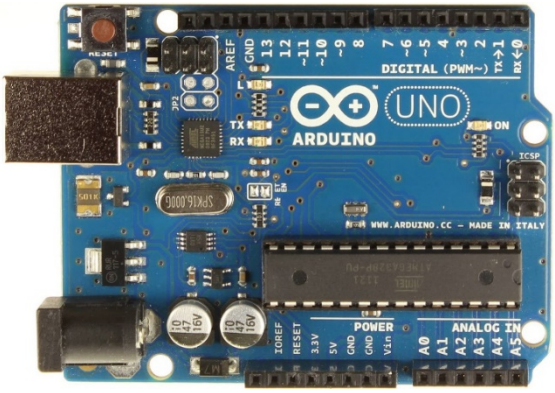
Obr. 2-1 Arduino IDE

2.1 Desky Arduino

Vzhledem k množství desek, které již existují, jsou uvedeny pouze ty, které jsou nejpoužívanější a jsou vhodné pro využití v domací automatizaci.

1) Arduino UNO (rev3)

V současné době existuje již třetí verze této desky. Vzhledem ke svým univerzálním parametrům a nízké ceně je v dnešní době nejpoužívanější. (12)



Obr. 2-2 Arduino UNO (6)

Parametry:

MCU	ATmega328P
Pracovní napětí	5V
Vstupní napětí (doporučené)	7-12V
Počet Digitálních I/O pinů	14 (6 umožňují PWM výstup)
Počet analogových pinů	6
Proud v I/O pinech	20mA
Flash paměť	32KB (0,5KB využívá Bootloader)
Taktová rychlost	16 MHz

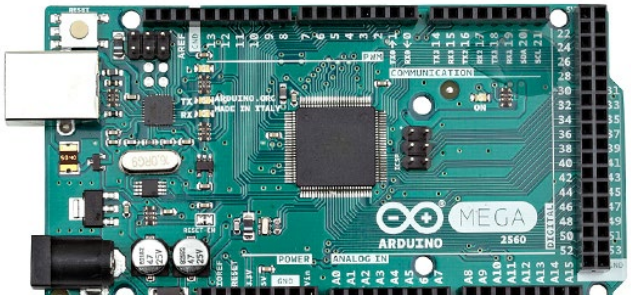
Deska obsahuje USB konektor, což umožňuje lehké nahrávání programů přímo z PC přes USB rozhraní a není potřeba mít programátor PIC.

Z 16 Digitálních I/O pinů nejsou všechny využitelné jako I/O piny, protože piny 0 a 1 mají vyhrazenou funkci pro příjem (pin 0) a posílání(pin 1) dat po sériové lince.

6 analogových vstupů měří v rozsahu 0 až 5V v rozlišení 10bitů (1024 hodnot). (13)

2) Arduino MEGA 2560

Pro využití v projektech, které potřebuji větší množství pinů, je hojně používaná deska Arduino MEGA 2560. (12)



Obr. 2-3 Arduino MEGA 2560 (6)

Parametry:

MCU	ATmega2560
Pracovní napětí	5V
Vstupní napětí (doporučené)	7-12V
Počet Digitálních I/O pinů	54 (15 umožňují PWM výstup)
Počet analogových pinů	16
Proud v I/O pinech	20mA
Flash paměť	256KB (8KB využívá Bootloader)
Taktová rychlost	16 MHz

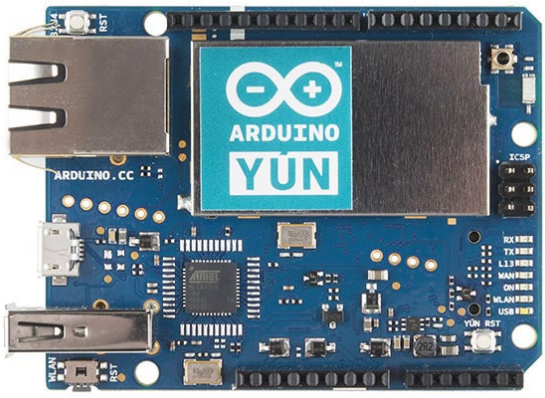
Z parametrů je zřejmé, že oproti desce Arduino UNO je využit silnější mikročip, a proto je zde větší množství pinů a větší paměť.

Dalším rozdílem je, že tato deska umožňuje sériovou komunikaci s třemi dalšími zařízeními pomocí pinů 14 až 19.

Díky většímu množství pinů a možnosti komunikovat s dalšími třemi zařízeními je tato deska vhodná jako centrální jednotka, která komunikuje s třemi „podřízenými“ deskami. (12)

3) Arduino YÚN

Pro domácí automatizaci je také vhodná deska Arduno YÚN. Tato deska se více odlišuje od předchozích, protože obsahuje ethernet konektor, Wi-Fi čip a také microprocesor, na kterém běží linux. Toto umoňuje propojit Arduino s domácí sítí a ovládat ho pomocí telefonu či tabletu. (12)



Obr. 2-4 Arduino YÚN (6)

Parametry (MCU):

MCU	ATmega32U4
Pracovní napětí	5V
Vstupní napětí (doporučené)	5V
Počet Digitálních I/O pinů	20 (7 umožňují PWM výstup)
Počet analogových pinů	12
Proud v I/O pinech	40mA
Flash paměť	32KB (4KB využívá Bootloader)
Taktová rychlost	16 MHz

Parametry (Mikročip s linuxem):

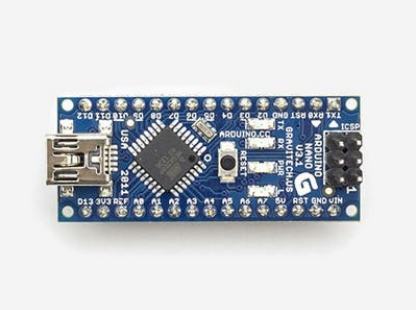
MCU	Atheros AR9331
Pracovní napětí	3,3V
RAM	64 DDR2
Flash paměť	16KB
Taktová rychlost	400 MHz

Díky přítomnosti Wi-Fi je velice praktická pro síť, kdy chceme využívat bezdrátovou komunikaci mezi Arduino deskami s touto deskou jako centrální a ostatní desky mohou být jiného typu s Wifi modulem.

Na linuxovém mikroprocesoru může běžet webový server, který může fungovat jako rozhraní pro uživatele na domácí sítě, který bude přístupný z různých zařízení.

4) Arduino NANO

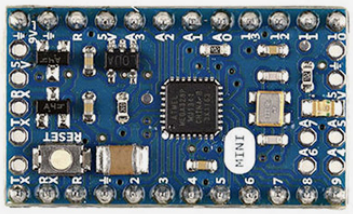
Tato deska je menší variantou desky Arduino UNO. Je na ní stejný mikročip ATmega 328P, je menší, a proto na této desce nenajdeme některé komponenty jako na Arduino UNO. Díky menším rozměrům je vhodná do hotových řešení, kde je malá velikost žádoucí. (12)



Obr. 2-5 Arduino NANO (6)

5) Arduino MINI

Tato deska je ještě menší než Arduino NANO. Vzhledem k tomu postrádá tato deska i USB rozhraní, a proto je třeba k jejímu programování využít externí převodník. Díky malým rozměrům je deska vhodná pro využití ve vypínačích, dálkových ovladačích apod. (12)



Obr. 2-6 Arduino MINI (6)

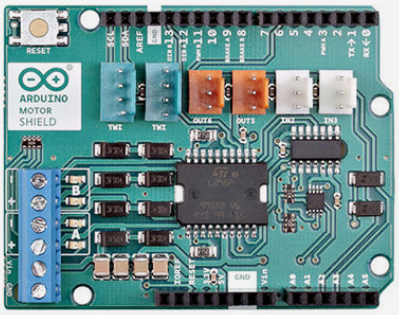
Existuje velké množství jiných Arduino desek či klonů od jiných firem. Tyto byly vybrány jako nejpraktičtější a hodící se pro využití v domácích automatizacích.

2.2 Arduino shieldy

Pro rozšíření základních desek se využívají shieldy, které se připojují k Arduino deskám. Tyto shieldy jsou svými rozměry vhodné pro připojení k desce Arduino UNO či deskám, které mají stejně vzdálené konektory.

1) Motor shield

Tento shield slouží k ovládání dvou DC motorů do napětí až 12V a proudu 2A na kanál (4A max s externím zdrojem napětí). Je založen na L298, což je H-můstek, který slouží k ovládání DC motorů, relé či krokových motorů. (12)

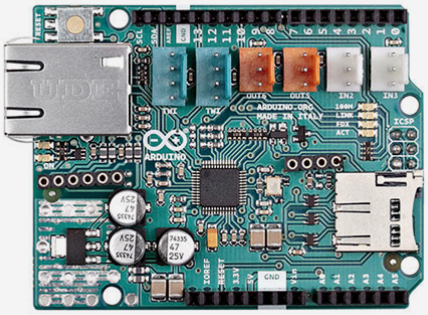


Obr. 2-7 Motor shield (6)

2) Ethernet shield v2

Tento shield rozšiřuje možnosti Arduino o ethernetové připojení k síti. Pokud je tedy potřeba větší množství pinů než nabízí Arduino YÚN, tak kombinace Arduino MEGA a tohoto shieldu je dobrá volba.

Shield je schopen připojení o rychlosti 10/100Mb a obsahuje slot pro SD kartu, na kterou lze nahrát kódy webových stránek. (12)

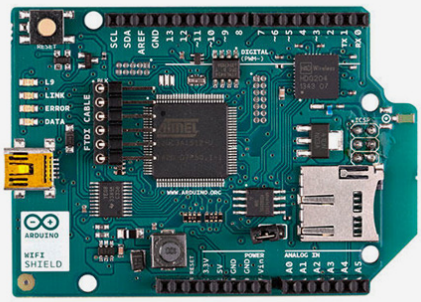


Obr. 2-8 Ethernet shield v2 (6)

3) WiFi shield

Obdobou ethernet shieldu je WiFi shield, který umožňuje bezdrátovou komunikaci přes WiFi síť.

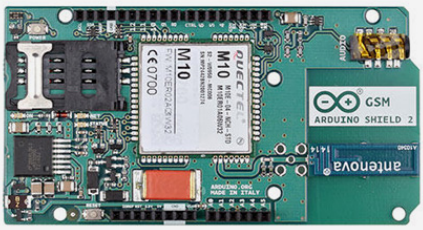
Tento shield je schopen komunikovat na sítích 802.11b/g, obsahuje slot pro SD kartu, FTDI konektor pro debugging samotného WiFi shieldu přes seriovou linku a Mini USB konektor pro update firmware shieldu. (12)



Obr. 2-9 Wi-Fi shield (6)

4) GSM shield v2

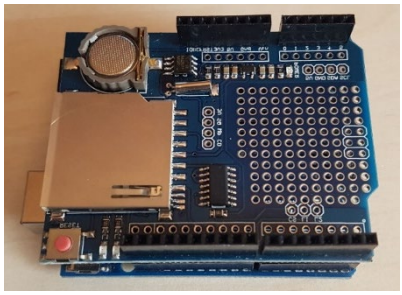
Tento shield umožní připojení Arduino k GPRS síti. Je možné připojení k internetu, příjmání/odesílání SMS a také příjmání/odesílání hovorů s využitím reproduktoru/mikrofonu připojeného k desce.



Obr. 2-10 GSM shield v2 (6)

5) Záznamový shield

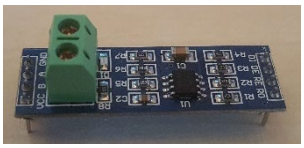
Tento shield je vybaven čtečkou SD karet a čipem reálných hodin a je tedy používán zejména k ukládání dat na SD kartu, kdy je také potřeba zaznamenat aktuální čas. Ale lze také použít k čtení dat z SD karty.



Obr. 2-11 Záznamový shield

6) Převodník RS485 – TTL

Jedná se o malou desku, která umožňuje komunikaci Arduina se zařízeními na sběrnici RS485 či propojení více Arduino sběrnicí RS485. S využitím této desky se dá snadno vytvořit komunikační sběrnice pro Arduino.



Obr. 2-12 Převodník RS485-TTL

Stejně jako u Arduino desek i u shieldů vzniklo nemalé množství klonů či jiných shieldů sloužících k jiným účelům. Existují shieldy pro 3D tiskárny, s displejem či senzory.

3 Návrh chytré domácnosti s využitím Arduino

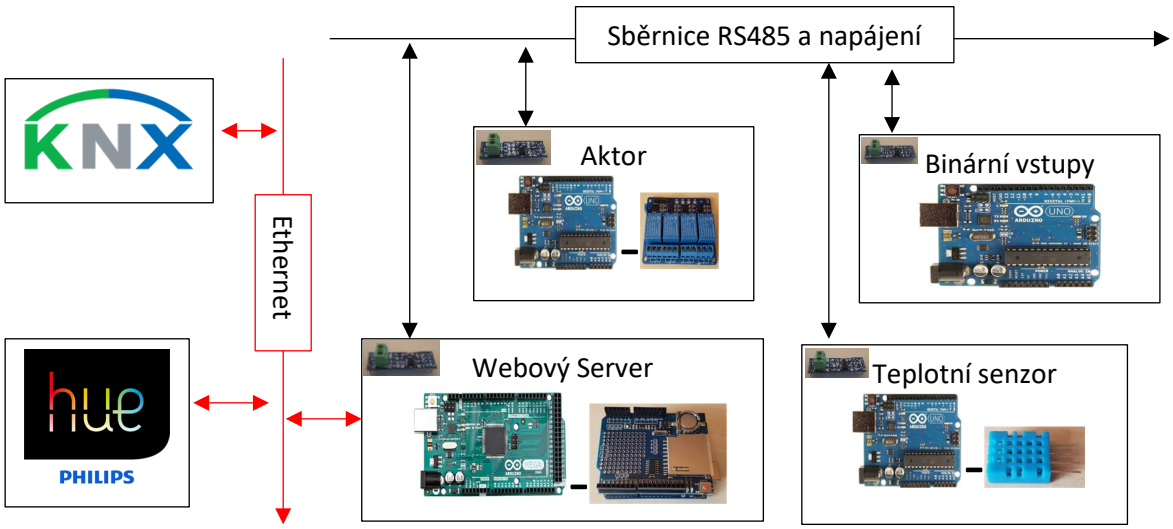
Vzhledem k tomu, že řešení Philips Hue nepokrývá všechny potřeby chytré domácnosti, bude v následující kapitole představeno řešení umožňující rozšíření o další prvky využívající platformu Arduino. V případě standardu KNX je důvodem pro využití platformy Arduino vysoká cena KNX zařízení. S využitím Arduino budeme schopni dosáhnout podobných výsledků, ovšem za výrazně nižší cenu.

3.1 Architektura řešení

Systém, který byl využit v řešení byl velice podobný řešení KNX. Tedy sběrnicové řešení, které využívá celkem 4 vodiče, dva vodiče jako napájení jednotlivých zařízení a dva vodiče pro komunikaci.

Jako sběrnice pro toto řešení byl využit standard RS485. Vzhledem k tomu, že Arduino nelze přímo napojit na sběrnici RS485 bylo potřeba mezi každým Arduino a sběrnicí převodník RS485 na TTL.

Zjednodušené blokové schéma řešení je na obr. 3-2.



Obr. 3-2 Blokové schéma řešení

Díky této architektuře bylo možné, aby všechna zařízení byla napájena z jednoho zdroje napětí, sdílela stejnou zem a možnost komunikace mezi všemi zařízeními mezi sebou.

3.2 Návrh komunikačního protokolu

Pro toto řešení bylo zvoleno, že posílaná zpráva bude mít fixní délku 9 bytů. Fixní délka zprávy byla zvolena proto, aby funkce pro posílání a přijímaní zpráv byly méně nachylné na chybovost.

Zpráva obsahuje následující byty:

1) 0x02 – Start byte, aby zařízení poznalo, že má začít zaznamenávat přenos;
2) Adresát – Adresa cílového zařízení, kterému je zpráva určena;
3) Odesilatel – Adresa zařízení, které poslalo zprávu;
4) Proměnná – Na této pozici jsou dvě varianty, buď se zde nachází adresa pinu, která se má aktivovat nebo se zde nachází typ proměnné, která je posílána;
5) Jednotky – Číslice na řádu jednotek;
6) Desítky – Číslice na řádu desítek;
7) Stovky – Číslice na řádu stovek;
8) CheckSum – Nejnižší byte kontrolního součtu;
9) 0x03 – Stop byte, který slouží k tomu aby zařízení poznalo, že se jedná o konec přenosu.

Po odeslání zprávy čeká odesílající zařízení na potvrzující zprávu o tom, že zpráva byla přijata. Pokud odesílatel nedostane potvrzující zprávu, dochází k opakovanému zaslání zprávy, k dalšímu opakovanému zaslání už nedochází.

3.3 Typy zpráv

Protokol umožňuje posílání různých typů zpráv, níže jsou 3 základní typy posílaných zpráv.

1) Potvrzovací zpráva

Ve chvíli, kdy zařízení přijme zprávu, odesílá potvrzovací zprávu odesílateli zprávy o přijetí zprávy. Proměnná zprávy je hodnota F (Feedback) a v parametrech se nachází nuly.

Start (0x02)	1 (0x31)	2 (0x32)	F (0x46)	0 (0x30)	0 (0x30)	0 (0x30)	CheckSum	End (0x03)

Obr. 3-3 Zpráva zpětné vazby

2) Zpráva obsahující analogové hodnoty

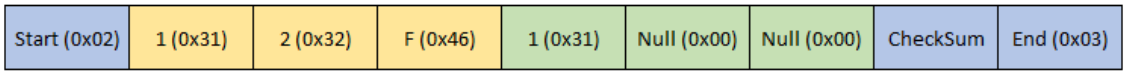
Součástí některých modulů je také možnost zasílání analogových hodnot fyzikálních veličin. Podle zasílané veličiny se zvolí proměnná, teplota T, vlhkost H, jas B. Je třeba zvolit, zda je hodnota určena specifickému zařízení nebo je určena pro všechna zařízení na sběrnici. V případě, že je určena pro všechna zařízení na sběrnici, tak bude byte s adresátem nulový (0x00).

Start (0x02)	1 (0x31)	2 (0x32)	T (0x54)	0 (0x30)	2 (0x32)	0 (0x30)	CheckSum	End (0x03)

Obr. 3-4 Zpráva obsahující naměřenou teplotu

3) Záslání akčního zásahu

Některé moduly jsou kontrolovatelné ostatními moduly a je tedy třeba mít možnost ovládat hodnoty na jednotlivých pinech. V tom případě je na pozici proměnné adresa pinu na cílovém zařízení. Další tři byty zprávy slouží k upřesnění typu akčního zásahu. Taková zpráva slouží pro zapnutí/vypnutí zařízení, změny směru pohybu pohonu či rozsvícení/zhasnutí osvětlení.



Obr. 3-5 Zpráva obsahující akční zásah

3.4 Příklady zpráv

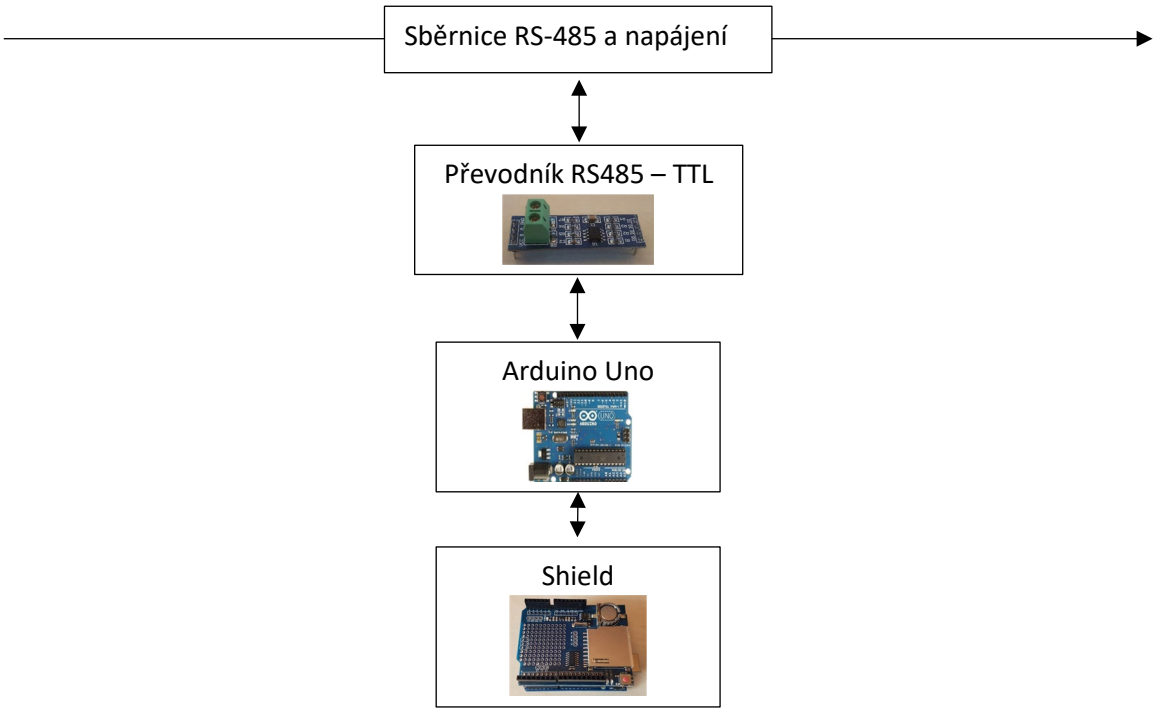
Řešení využívá pouze tři základní typy zpráv, ale vzhledem k tomu, že tyto typy zpráv jsou navrženy pro univerzální použití, tak z těchto zpráv lze vytvořit mnoho různých zpráv. Tabulka 2 obsahuje přehled zpráv posílaných po sběrnici, celkem se jedná o 9 zpráv.

Tabulka 2 Zprávy posílané po sběrnici

Typ zprávy	Start Byte	Adresát	Odesilatel	Proměnná	Jednotky	Desítky	Stovky	CheckSum	End Byte
Zpětná vazba	Start (0x02)	1 (0x31)	2 (0x32)	F (0x46)	0 (0x30)	0 (0x30)	0 (0x30)	CheckSum	End (0x03)
Teplota	Start (0x02)	1 (0x31)	2 (0x32)	T (0x54)	0 (0x30)	2 (0x32)	0 (0x30)	CheckSum	End (0x03)
Vlhkost	Start (0x02)	1 (0x31)	2 (0x32)	H (0x54)	3 (0x33)	5 (0x35)	0 (0x30)	CheckSum	End (0x03)
Jas	Start (0x02)	2 (0x32)	1 (0x31)	B (0x54)	1 (0x31)	2 (0x32)	0 (0x30)	CheckSum	End (0x03)
Vypnutí	Start (0x02)	1 (0x31)	2 (0x32)	1 (0x31)	0 (0x30)	Null (0x00)	Null (0x00)	CheckSum	End (0x03)
Zapnutí	Start (0x02)	1 (0x31)	2 (0x32)	5 (0x35)	1 (0x31)	Null (0x00)	Null (0x00)	CheckSum	End (0x03)
PWM 50%	Start (0x02)	1 (0x31)	2 (0x32)	5 (0x35)	H (0x48)	Null (0x00)	Null (0x00)	CheckSum	End (0x03)
PWM 25%	Start (0x02)	1 (0x31)	2 (0x32)	1 (0x31)	Q (0x51)	Null (0x00)	Null (0x00)	CheckSum	End (0x03)
Změnit hodnotu na pinu	Start (0x02)	1 (0x31)	2 (0x32)	1 (0x31)	T (0x54)	Null (0x00)	Null (0x00)	CheckSum	End (0x03)

3.5 Navržené moduly

Základ všech modulů byl navržen stejně, Arduino napojené díky převodníku RS485 - TTL na sběrnici. K tomu je na Arduino napojen jeden či více shieldů, dle potřeb daného modulu. Byly využity různé typy shieldů: námi vytvořený komunikační, záznamový a ethernetový.



Obr. 3-6 Obecné schéma modulů Arduino Řešení

Modul Binárních vstupů

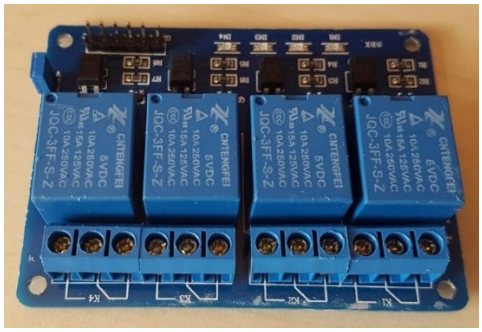
Toto zařízení je základní ovládací prvek řešení. Základem je deska Arduino Uno, která je napájena ze sdíleného zdroje a komunikuje s ostatními zařízeními na sběrnici. Díky univerzálnímu kódu tohoto zařízení je možné používat tento modul jako vypínač, který posílá předdefinované typy zpráv. Odeslání zprávy může být iniciováno stiskem tlačítka či binárním vstupem na pinu, například z IR senzoru.

Požadované zprávy:

- Změnit hodnotu pinu (byte 0x54);
- Nastavit pin HIGH (byte 0x31);
- Nastavit pin LOW (byte 0x30);
- Nastavit PWM na hodnotu 50% (byte 0x48);
- Nastavit PWM na hodnotu 25% (byte 0x51).

Spínací aktor

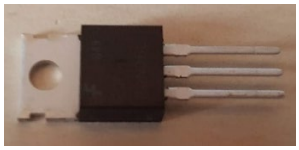
Toto zařízení je základní akční prvek řešení. Základem je deska Arduino Uno, která je napájena ze sdíleného zdroje a komunikuje s ostatními zařízeními na sběrnici. Arduino ovládá relé (obr. 3-4), která spínají zařízení. Pro účely této práce jsou na Arduino napojeny pouze 2 relé.



Obr. 3-7 Komerčně prodávaná relé deska

Spínací aktor – stmívací

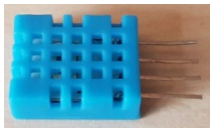
Základem je deska Arduino Uno, která je napájena ze sdíleného zdroje a komunikuje s ostatními zařízeními na sběrnici. Pomocí PWM regulace je možno regulovat jas ovládaného osvětlení. PWM je realizováno pomocí PWM výstupů Arduino a MOSFET tranzistoru. (obr. 3-8)



Obr. 3-8 MOSFET tranzistor 30N06L

Senzor – teplota a vlhkost

Základem tohoto senzoru je opět deska Arduino uno a senzor teploty a vlhkosti DHT11 (obr. 3-9). V nastavených intervalech zasílá naměřenou teplotu a vlhkost. Tento senzor má také možnost zasílat akční zásahy při překročení určitých hodnot.



Obr. 3-9 Senzor DHT11

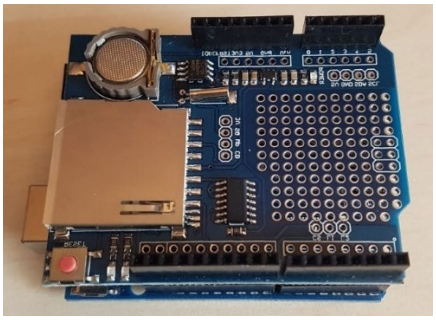
Senzor - jas

Základem tohoto senzoru je opět deska Arduino uno. V nastavených intervalech zasílá na řídící jednotku naměřenou intenzitu jasu v rozmezí 0 až 100 %. Senzor může také sám poslat akční zásah při překročení limitních hodnot.

Záznamové zařízení

Úkolem tohoto zařízení je sledovat veškerou komunikaci probíhající na sběrnici a ukládat zprávy do textového souboru na SD kartu. Ukládá se datum a čas přijetí zprávy. Proto musí být na toto zařízení připojen také modul reálného času. V nastavení modulu se dá určit, zda

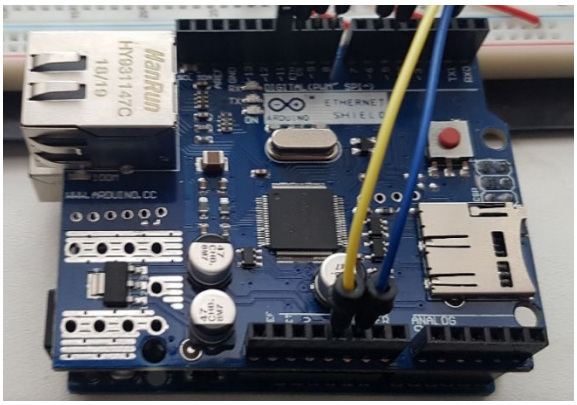
se mají ukládat zprávy ze senzorů. Pro tyto účely je využit komerčně prodávaný shield. (obr. 3-10)



Obr. 3-10 Komerčně prodávaný záznamový shield

Webový server

Vzhledem k náročnosti tohoto modulu realizováno s Arduino MEGA. Tento modul slouží jako spojení mezi řešeními a Arduino řešením. Zařízení na společné síti mohou zpřístupnit webové stránky hostované tímto zařízením. Zdrojový kód webových stránek se nachází na SD kartě, a proto při změně webu není potřeba měnit kód samotného Arduino a stačí pouze změnit obsah SD karty. Pro účely webového serveru bude využit komerčně prodávaný ethernet shield (obr. 3-11).



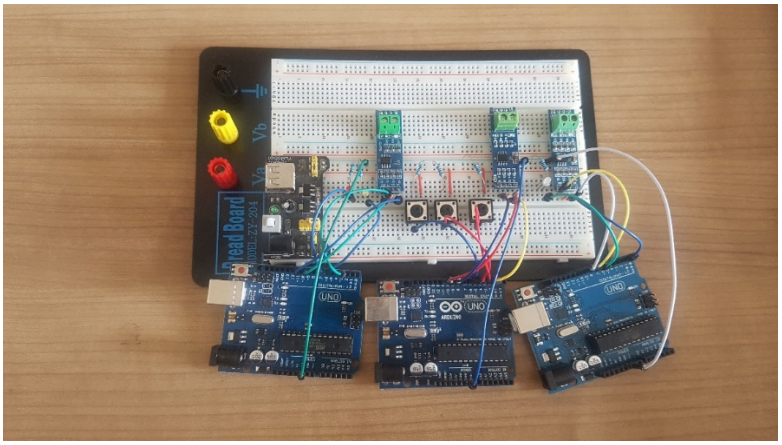
Obr. 3-11 Komerčně prodávaný ethernet shield

3.6 Řešení na nepájivém poli

V rámci testování byly jednotlivé moduly sestaveny na nepájivém poli a otestována jejich funkčnost. Kromě ethernet shieldu a záznamového shieldu nebyl využit žádný komerčně prodávaný shield a jednotlivé moduly byly sestaveny ze základních komponent jako relé, MOSFET či různé senzory. Komunikace vždy probíhala mezi jedním zařízením, které

figurovalo jako odesílatel a jedno jako příjemce zprávy. Všechna zařízení fungovala bezchybně.

Jako finální test realizace bylo na nepájivém poli sestaveno řešení sestávající se ze 3 zařízení Arduino, kdy jedno Arduino funguje jako vypínač a ovládá dvě zařízení, které fungují jako spínače. Pro účel realizace na nepájivém poli byly nahrazeny ovládané zařízení LED, které mohou simulovat zapínání zařízení či pohonů. Výsledná sestava je na obr. 3-12.



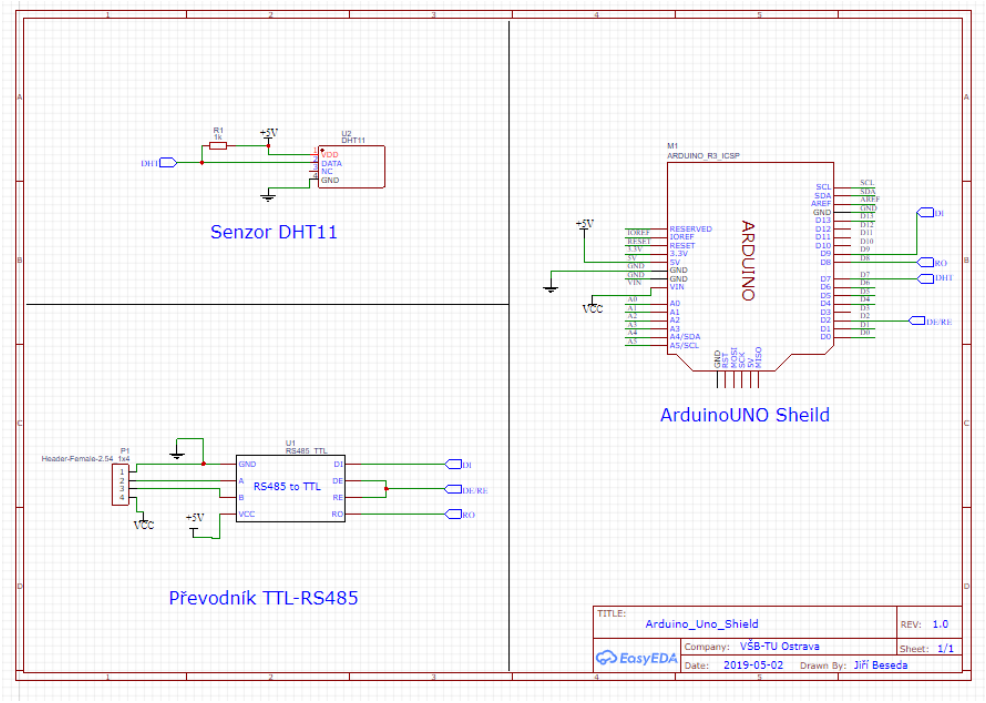
Obr. 3-12 Řešení na nepájivém poli

4 Realizace Arduino řešení

Pro napojení Arduino UNO na sběrnici byl navržen shield, který obsahuje převodník RS485 na TTL a umožňuje jednoduché napojení na sběrnici. Také je připraven k osazení senzorem DHT11 či DHT22 pro měření teploty a vlhkosti. Pro návrh shieldu byl použit cloudový nástroj a 10 kusů tohoto shieldu bylo objednáno u komerčního výrobce.

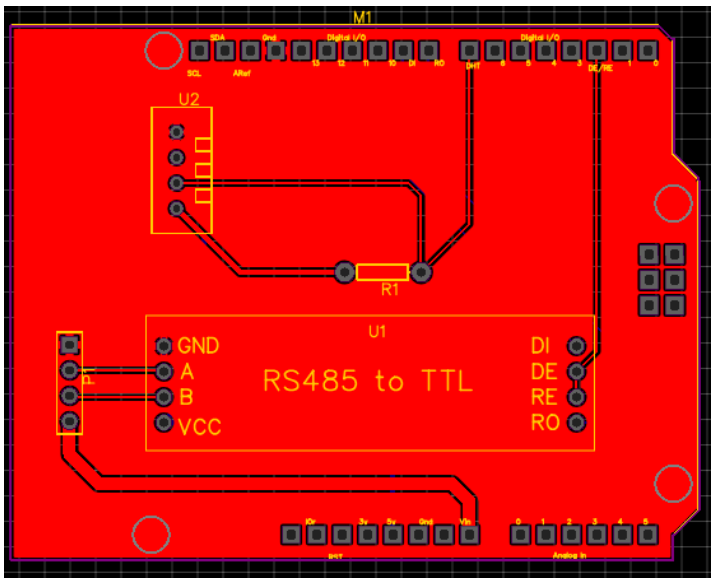
4.1 Návrh rozšiřující shieldu pro moduly

Návrh desky probíhá ve dvou krocích. V prvním kroku návrhu se vytvoří dokumentace. Tedy dokument obsahující všechny prvky budoucí desky. Cloudový designer obsahuje velkou knihovnu možný prvků či hotových desek jako je námi použitý převodník RS485-TTL. V návrhu se umístí všechny prvky do logických částí a vytvoří všechny napojení. Výsledný dokument lze vidět na obr. 4-1.



Obr. 4-1 Dokumentace k Arduino UNO komunikačnímu shieldu

V druhém kroku se provádí už reálný návrh budoucí desky. Všechny prvky se umístí na svá místa a poté se navrhnou spojení. Výrobce umožňuje za stejnou cenu provést dvouvrstvé desky, toto ulehčuje návrh desky. V průběhu vytváření desky je třeba dodržet několik pravidel, zejména nemít kolmé rohy spojení a zvolit správnou tloušťku podle předpokládaných hodnot napětí a proudů. Výsledný návrh lze vidět na obrázku. Lze na něm vidět různé tloušťky spojení a také že většinu desky zabírá plocha, která slouží jako zem a také k případnému chlazení desky.



Obr. 4-2 Návrh Arduino UNO shieldu

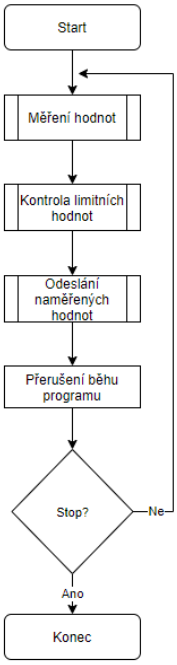
Tento návrh lze exportovat jako Gerber soubor nebo, jak jsme účinili v našem případě, objednat. Výrobce vyrábí levné prototypové desky za velice příznivé ceny. Po objednání desky může člověk sledovat v jaké fázi výroby se objednávka aktuálně nachází, malé objednávky jsou většinou hotovy do 24 hodin od objednání. Doručení je podle typu dopravy od 3 - 5 pracovních dní do několika týdnů v případě levnější dopravy. V našem případě byla zvolena rychlejší doprava z důvodu slevy na první objednávku. Desky dorazily během 5 dnů od objednání, všechny v pořádku, poté došlo k jejich zkompletování.

4.2 Prvky řešení

Navržené moduly byly zkompletovány a zapojeny do sběrnice. Jako propojovací kabel byl vybrán J-Y(ST)Y 2x2x0,8, který obsahuje 4 vodiče a je tedy ideální volbou.

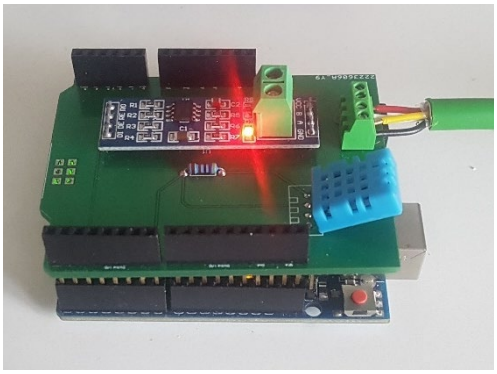
Senzor – Teplota, Vlhkost

Před použitím tohoto modulu je třeba nahrát do Arduino kód, ve kterém se nastaví adresa zařízení, limitní hodnoty, po jejichž překročení se má posílat akční zásah na jiná zařízení, výběr připojených senzorů a časové rozestupy měření. Hlavní smyčka samotného algoritmu (obr. 4-3) obsahuje odeslání naměřených hodnot a kontrolu překročení limitů. Tyto dva úkony se opakují v nastavené četnosti.



Obr. 4-3 Algoritmus modulu měřící teplotu a vlhkost

V případě realizovaného řešení, je na jeden modul napojen senzor teploty a vlhkosti DHT11. Byly nastaveny limitní hodnoty a četnost měření byla nastavena na každé 4 sekundy. Samotný modul lze vidět na obr. 4-4.

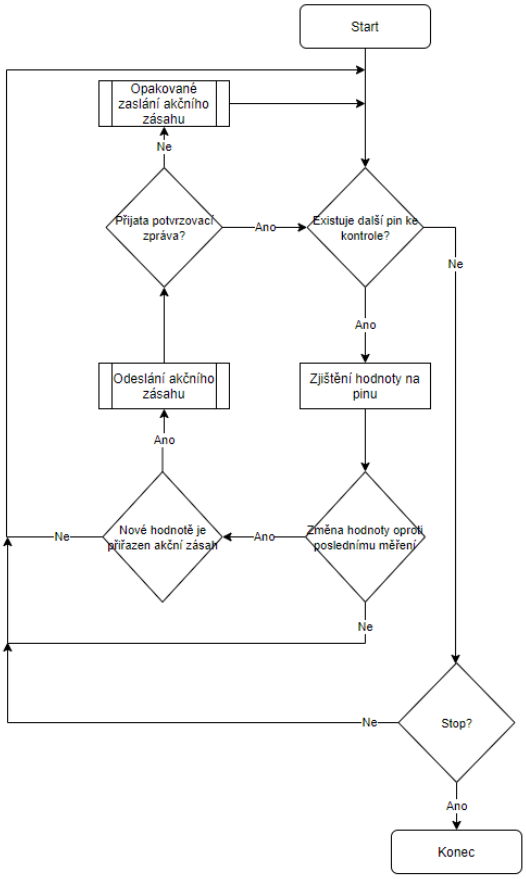


Obr. 4-4 Modul měření teploty a vlhkosti osazeny senzorem DHT11

Modul binárních vstupů

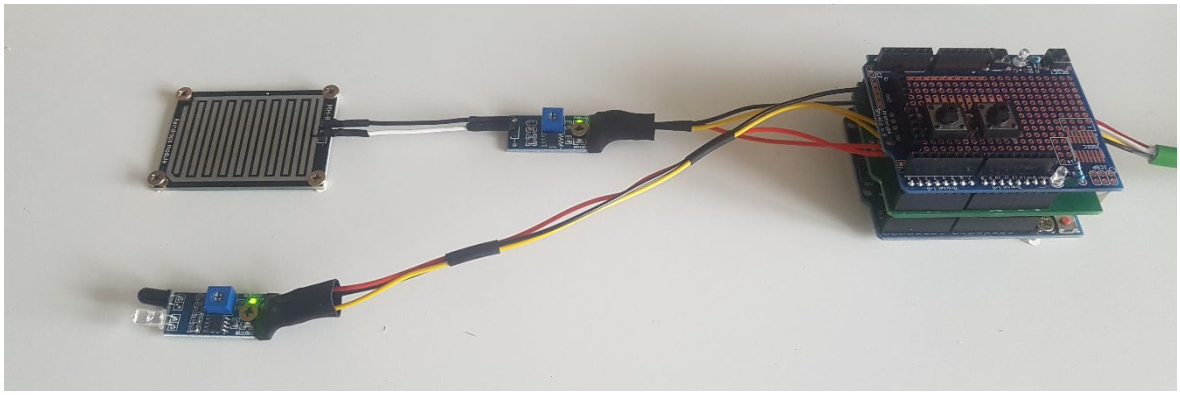
U tohoto modulu byla snaha o co největší univerzálnost, a proto existuje mnoho možností, jak jej nastavit. Na jeden pin vstupu lze nastavit několik akcí, je několik předdefinovaných zpráv, které lze odeslat a zprávy lze odesílat při sestupné či vzestupné hraně vstupu. Algoritmus tohoto modulu (obr. 4-6) má dvě části. V první části se kontroluje, zda došlo ke změně hodnoty na některém ze sledovaných pinů a v případě, že ano, tak dojde k odeslání přednastavené zprávy. V druhé části se přijímají zprávy a modul čeká na potvrzovací zprávu od cílového zařízení. V případě, že k potvrzení nedojde, nastane opakované odeslání

zprávy. Druhá zpráva je finální, po ní nenásledují další kontroly ani opakované odeslání zpráv.



Obr. 4-5 Algoritmus modulu binárních vstupů

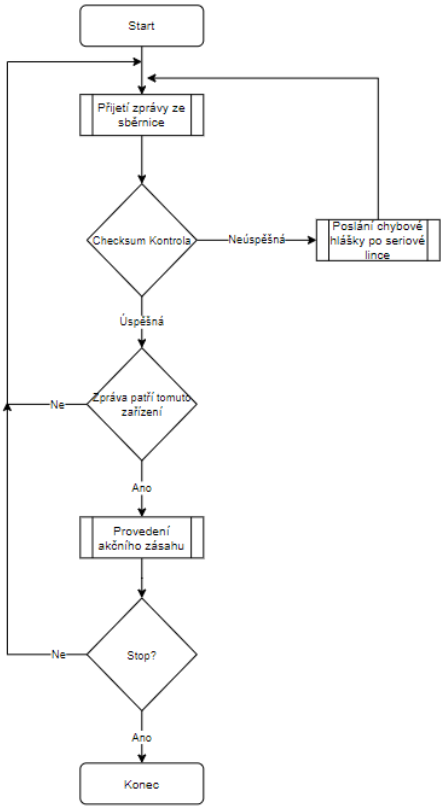
V realizovaném řešení byl k modulu připojen IR senzor, který v případě objektu v detekované oblasti odešle zprávu a další, když se objekt v oblasti už nacházet nebude. V praxi se dá tento senzor použít například jako součást alarmu. Dále byl připojen senzor sledující, zda se nachází v mokrém prostředí. Při namočení odesílá zprávu a po uschnutí také. V praxi se tento senzor dá použít pro detekci deště nebo úniku vody a v obou případech tak lze předcházet škodám způsobeným vodou. Kromě dvou senzorů byl také modul vybaven shieldem, na kterém se nachází dvě tlačítka pro odesílání zpráv. Vstupům byly přiřazeny různé přednastavené zprávy pro využití všech možností řešení. Využity byly pouze 4 piny: 5,6,7,12, další vstupy lze připojit na piny 3,4,10,11. Modul s připojenými senzory lze vidět na obr. 4-6.



Obr. 4-6 Modul binárních vstupů s připjeným IR senzorem, sensorem namočení a shieldem s dvěmi tlačítky.

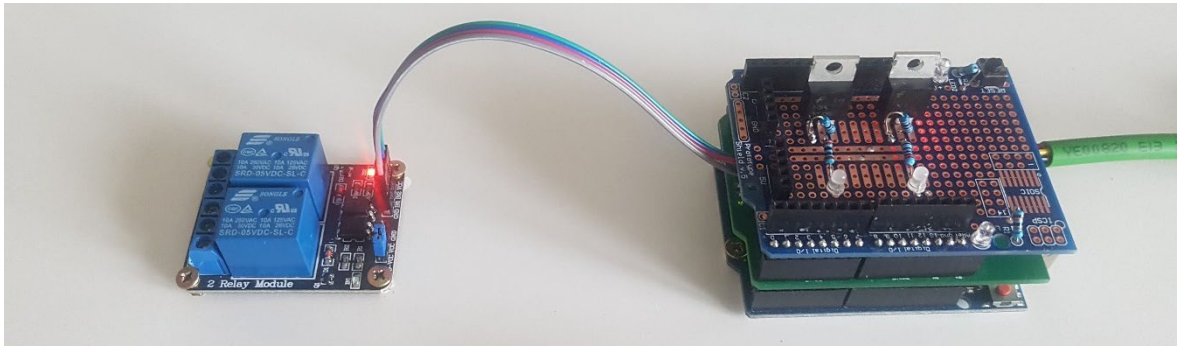
Spínací aktor

U tohoto modulu je opět zřejmá snaha o univerzálnost. Tento modul lze využít v diskrétním spínání hodnot na pinech nebo ve spojitých díky PWM. V nastavení modulu se určí adresy pinů a jejich výchozí hodnoty po zapnutí modulu. V algoritmu modulu (obr. 4-7) dochází po přijetí zprávy ke kontrolám a po zjištění, že zpráva je určena pro daný modul a kontrolní součet odpovídá kontrolnímu součtu ve zprávě, modul odesílá zpětnou vazbu odesílateli a poté provede požadovanou akci.



Obr. 4-7 Algoritmus spínacího aktoru

V realizovaném řešení jsou k modulu připojeny dvě relé. První relé napájí čerpadlo, které se spouští v případě, že senzor mokrého prostředí zaznamená vodu na svém povrchu a vypíná, když je senzor suchý. Druhé relé se spouští zmáčknutím tlačítka a není k němu nic připojeno. Dále se na modulu nachází shield, který simuluje binární výstup. Tato simulace je realizována dvěmi LED, které jsou spínány pomocí dvou MOSFET tranzistorů. První LED je ovládána IR senzorem a rozsvítí se na 50 % své maximální intenzity, když se před senzorem nachází předmět, a zhasne, když se před senzorem nenachází žádný předmět. Druhá LED pouze mění svůj stav po každém zmáčknutí tlačítka na vstupním modulu. Výstupy byly napojeny na 4 piny: 5, 6, 10, 11 a proto zbývají ještě další piny, které lze využít jako výstupy. Jediné omezení je, že v kódu je použit timer1 a z toho důvodu nelze piny 9 a 10 využít s PWM výstupem. Realizovaný modul s připojeným relé lze vidět na obr. 4-8.

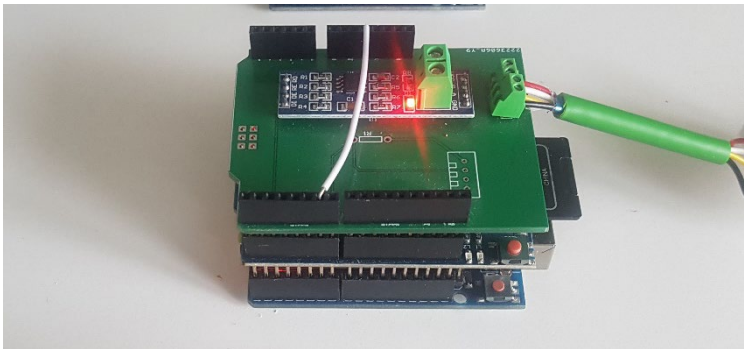


Obr. 4-8 Modul spínací aktor s připojeným relé a dvěmi LED spínanými pomoci MOSFET tranzistoru.

Záznamový modul

Tento modul, kromě námi vytvořeného komunikačního shieldu, využívá také shield obsahující čtečku karet a čip realných hodin. Toto zařízení přijímá všechny zprávy ze sběrnice a ukládá je na SD kartu včetně času, kdy došlo k přijetí zprávy. Při zapnutí modulu se zkontroluje hodnota na pinu 6, zda se má nastavit čas z kódu nebo použít dříve nastavený čas. Vzhledem k tomu, že v realizaci není baterie, tak je potřeba mít pin vždy nastaven na HIGH, aby došlo k inicializaci hodin. Přijaté zprávy lze zkontrolovat na PC v souboru log.txt, který se bude nacházet na kartě.

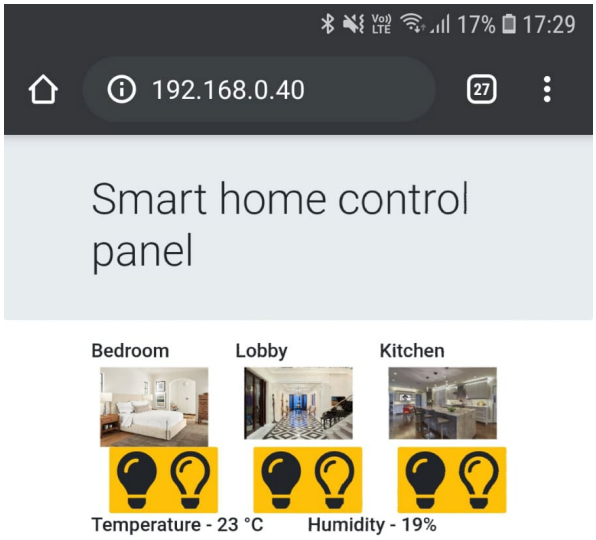
V realizaci se v shieldu nenachází baterie a je tedy při každém zapnutí nastaven čas a datum, který je nastaven v kódu jako výchozí. Z toho důvodu je pin 6 propojen s pinem 5V pro nastavení aktuálního času z kódu. Modul lze vidět na obr. 4-9.



Obr. 4-9 Záznamový modul

Webový server

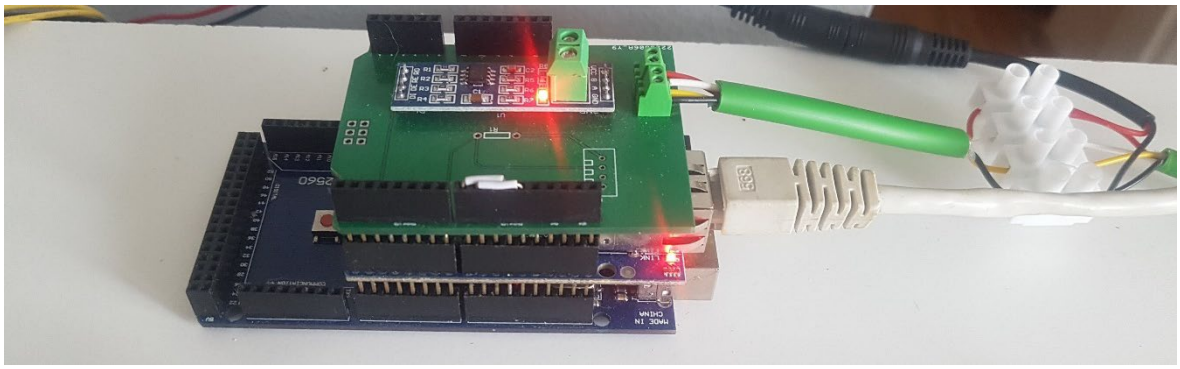
Vzhledem k tomu, že se jedná o webový server, tak pro správné fungování serveru je potřeba na SD kartu nahrát zdrojový kód webu, který se bude zobrazovat uživatelům, kteří se pokusí připojit na IP adresu serveru. Dále je také potřeba nastavit akce, které budou přiřazeny tlačítkům na webovém serveru. Co se týká algoritmu modulu, tak hlavním úkolem je zpracovávat požadavky uživatelů webu. Po zmáčknutí tlačítka na webu dochází k zavolání příslušné funkce v kódu modulu, a buď přepošle požadavek na jiné zařízení Arduino nebo na zařízení Philips Hue. Také přijímá hodnoty naměřené na senzorech a zobrazuje je na webu. Vizualizace přístupná z mobilního telefonu lze vidět na obr. 4-10.



Obr. 4-10 Webové stránky Arduino řešení

V realizaci je web pouze základní rozhraní ilustrující možnosti řešení. Obsahuje ovládání tří výstupů. První je světlo Philips Hue, druhé jedno z relé a třetí je LED. Dále se na webu zobrazuje aktuální teplota a vlhkost. Server má také pevně nastavenou IP adresu a je tedy nutné na routeru, kde se webový server nachází, rezervovat tuto IP adresu. Pevně

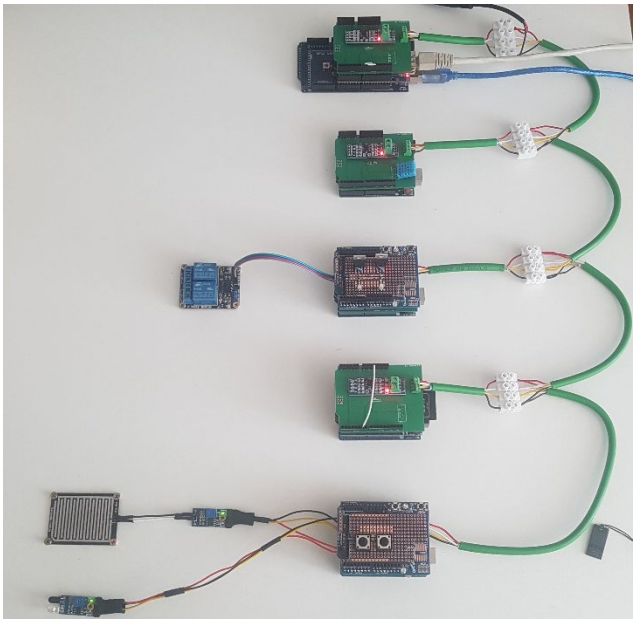
nastavená IP adresa umožňuje přístupovat k serveru z jiných zařízení, například KNX. Modul lze vidět na obr. 4-11.



Obr. 4-11 Modul webového serveru připojený na ethernet

4.3 Zkompletované řešení

Všechna zařízení byla naprogramována a zapojena do společné sběrnice umožňující napájení a vzájemnou komunikaci všech zařízení (obr. 4-12). Dále proběhlo napojení na komerční systémy.



Obr. 4-12 Zkompletované řešení

Napojení na Philips Hue

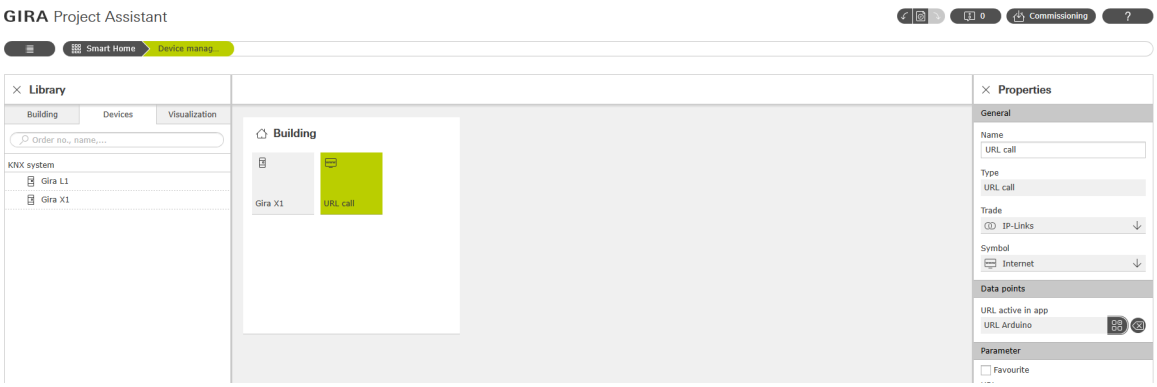
Napojení na Philips Hue je využívá Philips Hue API a umožňuje ovládat zařízení Philips Hue z webového serveru Arduino. Samotné nastavení API bylo popsáno v kapitole 1.2.2. Komunikace je umožněna směrem z Arduino řešení do Philips Hue řešení. V případě

rozšíření kódu webového serveru či vytvoření nového prvku Arduino řešení, by bylo možné vytvořit oboustrannou komunikaci.

Napojení na KNX

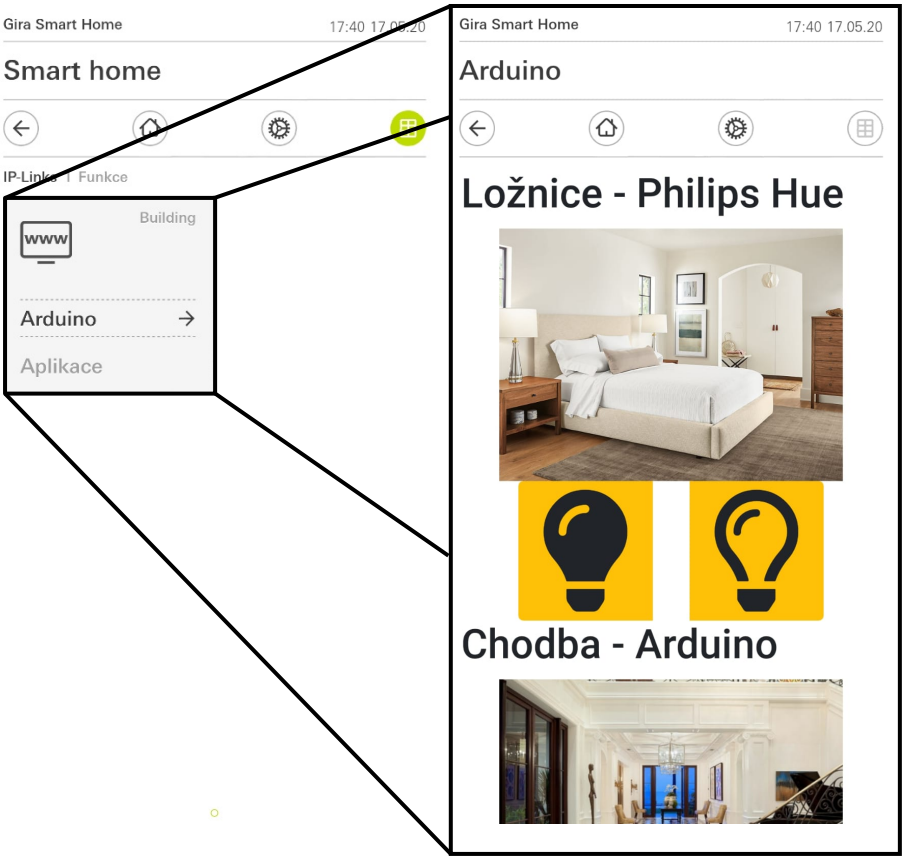
Pro napojení Arduino řešení na řešení KNX bude využit dříve zmíněný server X1 od firmy Gira. Toto zařízení umožní zobrazení webové stránky Arduino serveru ve vizualizaci KNX. Z toho vyplývá omezení, že komunikace je pouze jednosměrná z KNX vizualizace do Arduino řešení. Toto omezení plyne pouze z možností zařízení X1, v případě využití KNX modulu binárních vstupů lze mít obousměrnou komunikaci.

Gira X1 se nastavuje pomocí programu Gira Project Assistant, který je v současnosti ve verzi 4.2. Tento program slouží k nastavení všech funkcí X1. Funkce, která bude námi využita je zobrazování URL ve vizualizacích KNX. V programu se vytvoří místnosti, vkládají se KNX zařízení a také samotné akce dostupné ve vizualizacích. Po vytvoření proměnné stačí vytvořit budovu, přidat zařízení X1 a prvek vizualizace URL Call. V nastavení se poté přiřadí jako použitá proměnná námi dříve vytvořená proměnná s URL adresou webového serveru. Kompletní nastavení lze vidět na obr. 4-13.



Obr. 4-13 Nastavení X1 pro přístup k Arduino řešení

Po nahrání tohoto projektu do X1 lze vizualizace zobrazit přes aplikaci v tabletu nebo mobilním telefonu. Vzhled vizualizace je na obr. 4-14.



Obr. 4-14 Arduino web v prostředí Gira Smart Home

Závěr

V rámci přehledu bylo analyzováno komerční řešení od firmy Philips s obchodním názvem Philips Hue. Toto řešení se soustřeďuje na ovládání světel a bylo tedy vhodným kandidátem na rozšíření o možnost ovládat i jiná zařízení jako brány a žaluzie. Zařízení, která jsou standardně součastí řešení Philips Hue, jsou relativně drahá a vytvořené Arduino řešení se ukázalo jako levnější varianta prvků Philips Hue.

Druhým analyzovaným řešením byl standard KNX, který je používán jako profesionální řešení pro automatizaci budov. Zařízení využívající tento standard pokrývají všechny potřeby chytré domácnosti. Jedná se o spolehlivé řešení s životností v desítkách let. Nevýhodou je velmi vysoká cena, a proto je opět Arduino vhodným kandidátem, jak snížit cenu takového řešení.

V rámci návrhu řešení byl vypracován přehled různých Arduino desek, které jsou dostupné na trhu a také shieldů, které fungují jako rozšíření možností standardní desky Arduino o funkčnosti jako GSM komunikace a možnost připojení na ethernet.

Jako vhodná deska bylo vybráno univerzální Arduino Uno, pro více náročné moduly Arduino Mega. Byl navržen komunikační protokol využívající standard RS485 a naprogramován kód Arduino umožňující komunikaci. Tato komunikace byla poté otestována na nepájivém poli. Po úspěšném otestování komunikace mezi moduly na nepájivém poli, byl využit komerční výrobce tištěných spojů a byly navrženy a vyrobeny shieldy pro snadné napojení Arduino na sběrnici. Byly navrženy a naprogramovány moduly, které jsou vhodné jako prvky inteligentní domácnosti, webový server, modul vstupů a výstupů, senzor teploty a záznamové zařízení. Po vyrobení komunikačního shieldu pro moduly a nahrání potřebných kódů, byly moduly zapojeny do sběrnice, která slouží pro komunikaci a také jako napájení modulů. Celé toto řešení bylo rozšířeno o vstupní a výstupní periferie simulující chytrou domácnost, tlačítka, světla, senzory a relé. Celé řešení funguje bezchybně a po rozšíření o více modulů je vhodné jeho využití pro inteligentní dům.

Tato práce ukazuje, že komerčně prodávané produkty se dají rozšířit o námi navržené zařízení. Tato řešení se poté dají zkombinovat do jednoho řešení. Docílíme tím buď doplnění chybějících prvků v komerčních řešení jako jsme my ukázali na příkladu Philips Hue, nebo finančí úspory při výrobě zařízení místo jejich koupě v případě zařízení KNX. Možnost propojení je umožněna zejména z toho důvodu, že existuje velké množství

komerčních řešení a výrobci přizpůsobují své výrobky tak, aby se daly připojit na řešení jiných značek. V dnešní době roste popularita inteligentních domácností a dá se tedy čekat, že v budoucnu vznikne mnoho dalších produktů od různých firem a jejich cena pravděpodobně klesne. Aktuálním problémem je zabezpečení chytrých domácností. Některé výrobky jsou snadno napadnutelné a umožní přístup do ovládání chytré domácnosti zvenčí neautorizovanou osobou. Arduino řešení proto musí mít sběrnici umístěnu tak, aby nebyla dostupná zvenčí.

Možným rozšířením řešení by mohlo být vytvoření dalších typů modulů doplněných o logické operace. Tento modul by odesílal zprávy na základě dříve poslaných zpráv. Modul s PID regulací by určitě také našel využití v chytré domácnosti. Případně by bylo možné vytvořit specializované moduly pro jednotlivé úkony, řízení světel, motorů či stmívače. Dalším možným rozšířením řešení by mohlo být doprogramování komunikace, aby všechny zprávy posílané po sběrnici byly šifrované a minimalizovalo se tím pádem riziko napadení.

Bohužel finální kompletace řešení probíhala v průběhu COVID-19 pandemie, kdy většina obchodů byla zavřená a bylo obtížné sehnat další potřebné komponenty a musely být využity komponenty dříve nakoupené.

Použitá literatura

(1) *Loxone* [online]. Rakousko: LOXONE ELECTRONICS GMBH, 2020 [cit. 2020-05-17]. Dostupné z: loxone.com

(2) *KNX* [online]. Belgie: KNX, 2020 [cit. 2020-05-17]. Dostupné z: https://www.knx.org/

(3) Communication Media. *Knx.org* [online]. Brusel: KNX Association cvba, 2017 [cit. 2019-04-19]. Dostupné z: https://www2.knx.org/za/knx/technology/communication-media/index.php

(4) *Gira* [online]. Německo: Gira, 2020 [cit. 2020-05-17]. Dostupné z: www.gira.com

(5) Philips Hue - Architecture. In: *Meet Hue* [online]. Amsterdam: Philips Lighting, 2018 [cit. 2019-05-12]. Dostupné z: https://developers.meethue.com/wp-content/uploads/2018/02/Architecture_overview_entertainment.png

(6) *Home Assistant* [online]. Německo: Home Assistant, 2020 [cit. 2020-05-17]. Dostupné z: https://www.home-assistant.io/

(7) *IFTT* [online]. Spojene státy americké: IFTT, 2020 [cit. 2020-05-17]. Dostupné z: https://ifttt.com/

(8) Amazon Alexa. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2020-05-17]. Dostupné z: https://en.wikipedia.org/wiki/Amazon_Alexa

(9) Google Assistant. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2020-05-17]. Dostupné z: https://en.wikipedia.org/wiki/Google_Assistant

(10) Z-Wave. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2020-05-17]. Dostupné z: https://en.wikipedia.org/wiki/Z-Wave

(11) *Fibaro* [online]. Polsko: Fibar Group, 2020 [cit. 2020-05-17]. Dostupné z: https://www.fibaro.com/

(12) *Arduino* [online]. Italy: Arduino, 2019 [cit. 2019-05-12]. Dostupné z: https://www.arduino.cc/

(13) ARDUINO.CC. Https://www.arduino.cc/en/Main/Products. *Arduino.cc* [online]. 2017 [cit. 2017]. Dostupné z: https://www.arduino.cc/

(14) Plan for intelligent Future Safety: Building Control Systems. *Schneider Electric* [online]. France: Schneider Electric, 2018 [cit. 2019-04-19]. Dostupné z: http://download.schneider-

electric.com/files?p_enDocType=Catalog&p_File_Name=LSB02779_EN_KNX_Catalogue_11_2018%28web%29.pdf&p_Doc_Ref=LSB02779_EN

(15) MARGOLIS, Michael. *Arduino cookbook*. 2nd ed. Sebastopol, Calif.: O'Reilly, 2012. ISBN 978-1-449-31387-6.

(16) *Schneider Electric* [online]. Francie: F, 2020 [cit. 2020-05-17]. Dostupné z: www.se.com

Přílohy

A) HTML kód rozhraní webového serveru
B) Kód modulu binárních vstupů
C) Kód modulu spínacího aktoru

Příloha A – HTML kód rozhraní webového serveru

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.8.2/css/all.css" integrity="sha384-
oS3vJWv+0UjzBfQzYUhtDYW+Pj2yciDJxpsK1OYPAYjqT085Qq/1cq5FLXAZQ7Ay"
crossorigin="anonymous">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <meta charset="utf-8">
    <title>Smart home webpage</title>
    <script>
        strText = "";
        function SwitchHueLight(id, status) {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            strText = "&type=Hue&id=" + id + "&status=" + status + "&end=end";
            request.open("GET", "ajax_inputs" + strText + nocache, true);
            request.send(null);
        }

        function SwitchArduino(id, status) {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            strText = "&type=Arduino&id=" + id + "&status=" + status +
"&end=end";
            request.open("GET", "ajax_inputs" + strText + nocache, true);
            request.send(null);
        }

        function GetTempHumid() {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            request.onreadystatechange = function(){
                if (this.readyState == 4) {
                    if (this.status == 200) {
                        if (this.responseText != null) {
                            document.getElementById("temp_humid_data").innerHTML
= this.responseText;
                        }
                    }
                }
            }
            request.open("GET", "requestTempHumid" + nocache, true);
            request.send(null);
            setTimeout('GetTempHumid()', 5000);
        }
    </script>
    <style>
    </style>
</head>
<body onload="GetTempHumid();" style="background-color:#ffffff;">
<div class="jumbotron">
    <div class="container">
        <h1 class="display-3">Smart home control panel</h1>
    </div>
</div>
<div class="container">
    <!-- Example row of columns -->
    <div class="row">
        <div class="col-md-12">
            <h2>Ložnice - Philips Hue</h2>
            <div class="col-md-4">
                <img class="col-md-12"
src="https://s7d4.scene7.com/is/image/roomandboard/wyatt_438274_19e_g?hei=725&$st
```

```
r_g$" >
            </div>
            <div class="col-md-4">
                <div class="btn-group btn-group-justified col-md-12">
                    <div class="btn-group col-md-6">
                        <button type="button" class="btn btn-warning"
onclick="SwitchHueLight(1,220)"><i class="fas fa-lightbulb fa-6x"></i></button>
                    </div>
                    <div class="btn-group col-md-6">
                        <button type="button" class="btn btn-warning"
onclick="SwitchHueLight(1,0)"><i class="far fa-lightbulb fa-6x"></i></button>
                    </div>
                </div>
            </div>
            <div class="row">
                <div class="col-md-12">
                    <h2>Chodba - Arduino</h2>
                    <div class="col-md-4">
                        <img class="col-md-12"
src="http://interiordecoration.eu/wp-content/uploads/2015/01/8-Amazing-Entrance-
Lobby-Designs-1.png" >
                    </div>
                    <div class="col-md-4">
                        <div class="btn-group btn-group-justified col-md-12">
                            <div class="btn-group col-md-6">
                                <button type="button" class="btn btn-warning"
onclick="SwitchArduino(0,1)"><i class="fas fa-lightbulb fa-6x"></i></button>
                            </div>
                            <div class="btn-group col-md-6">
                                <button type="button" class="btn btn-warning"
onclick="SwitchArduino(1,0)"><i class="far fa-lightbulb fa-6x"></i></button>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-12">
                        <h2>Kuchyně - Arduino</h2>
                        <div class="col-md-4">
                            <img class="col-md-12" src="https://irp-
cdn.multiscreensite.com/a7ebb15f/dms3rep/multi/mobile/94a09b89-06a1-40a9-8bc6-
5b9658c5cdab.jpg" >
                        </div>
                        <div class="col-md-4">
                            <div class="btn-group btn-group-justified col-md-12">
                                <div class="btn-group col-md-6">
                                    <button type="button" class="btn btn-warning"
onclick="SwitchArduino(2,1)"><i class="fas fa-lightbulb fa-6x"></i></button>
                                </div>
                                <div class="btn-group col-md-6">
                                    <button type="button" class="btn btn-
warning"onclick="SwitchArduino(3,0)"><i class="far fa-lightbulb fa-
6x"></i></button>
                                </div>
                            </div>
                        </div>
                    </div>
                    <div class="row" id="temp_humid_data">
                        <div class="col-sm-12">
                            <h2>Temperature - N/A °C</h2>
                        </div>
                        <div class="col-md-12">
                            <h2>Humidity - N/A %</h2>
                        </div>
                    </div>
                </div>
</body>
</html>
```

Příloha B - Kód modulu binárních vstupů

```
#include <SoftwareSerial.h> // inicializijeme knihovnu
#define SSerialRX        8 //Serial Receive pin
#define SSerialTX        9 //Serial Transmit pin


#define SSerialTxControl 2   // RS modul pin 3
// vytvorime seriovy port na pinu 10 a 11 se jménem RS485Serial
SoftwareSerial RS485Serial(SSerialRX, SSerialTX); // RX, TX

int timeslot = 1;

//#################################################################################################################

// Unikátní pro každé arduino
byte myAddress = 0x31;
const int pinNumber = 6;
const int buttonPin[pinNumber] = {5, 5,6,6,7,12}; //available pins are 3,4,5,6,7,10,11,12,13
byte buttonAddress[pinNumber] = {0x31, 0x31, 0x32, 0x32, 0x33, 0x34};
int buttonDeviceAdress[pinNumber] = {0x32, 0x32, 0x32, 0x32, 0x32, 0x32};
byte buttonAction[pinNumber] = {0x31, 0x30, 0x48, 0x30, 0x54, 0x54}; // 0x30 turn off,0x31 turn on, 0x54 toggle, 0x48
for PWM half, 0x51 PWM quarter
int inputSendState[pinNumber] = {0, 1, 0, 1, 1,1};

//#################################################################################################################

int buttonState[pinNumber] = {1, 1, 1, 1, 1,1};
int lastButtonState[pinNumber] = {1, 1, 1, 1,1,1};
int lastState = 0;
int mappedValue = 0;
int programming = 0;
int deviceButtons = 0;
int deviceBinaryInput = 1;
byte ckSumValue = 0;
byte receivedData[10];
byte sendData[10];
char charRecived;
int serialCounter = 0;
char device;
int deviceint;
int msgconfirmation = HIGH;
int counterconfirmation = 0;
int msgsend = LOW;
int countersend = 0;

void setup()
{
 noInterrupts();         // disable all interrupts
 TCCR1A = 0;
 TCCR1B = 0;
 TCCR1B |= (1 << CS10) | (1 << CS11); //64 0,25s
 // TCCR1B |= (1 << CS12);            //256  1s
 //TCCR1B |= (1 << CS12) | (1 << CS10);  //1024 4s
 TIMSK1 |= (1 << TOIE1);  // enable timer overflow interrupt
 interrupts();           // enable all interrupts
 Serial.begin(9600);
 for (int i = 0; i < pinNumber; i++) {
  pinMode(buttonPin[i], INPUT_PULLUP);
  Serial.print("Pin: ");
  Serial.print(buttonPin[i]);
  Serial.println(" nastaven jako INPUT_PULLUP");
 }
 pinMode(SSerialTxControl, OUTPUT);
 digitalWrite(SSerialTxControl, 0);  // Povolí přijímaní na modulu RS485
```

```
 // nastartujeme software serial
 RS485Serial.begin(14400);
}
ISR(TIMER1_OVF_vect)       // interrupt service routine
{
 if (counterconfirmation == 2)
 {
  if (msgconfirmation == LOW)
  {
   counterconfirmation = 0;
   sendTelegram();
   Serial.println("ERROR: Telegram not cofirmed by receiver");
   msgconfirmation = HIGH;
   delay(50);
  }
  else
  {
   flushSendData();
  }
 }
 counterconfirmation = counterconfirmation + 1;
}
void loop()  {
 for (int i = 0; i < pinNumber; i++) {
  buttonState[i] = digitalRead(buttonPin[i]);
  // compare the buttonState to its previous state
  if (buttonState[i] != lastButtonState[i]) {
   // if the state has changed, increment the counter
   if (buttonState[i] == inputSendState[i] ) {
    Serial.print("zmačknuto ");
    Serial.println(buttonPin[i]);
    // if the current state is HIGH then the button went from off to on:
    ckSumValue = ckSum(buttonDeviceAdress[i], myAddress, buttonAddress[i], buttonAction[i], 0x30, 0x30);
    prepareData(buttonDeviceAdress[i], buttonAddress[i], buttonAction[i], 0x30, 0x30, ckSumValue);
    sendTelegram();
    msgconfirmation = LOW;
   } else {
    // if the current state is LOW then the button went from on to off:
   }
   // Delay a little bit to avoid bouncing
   delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastButtonState[i] = buttonState[i];
 }
 if (RS485Serial.available())
 {
  charRecived = RS485Serial.read(); //Přečte přijatý znak
  if (charRecived == 0x03) // když znak je 0x03 značí to konec vyslílání
  {
   serialCounter = 0;
   if (ckSumCheck() == true)
   {
    if (receivedData[1] == myAddress and receivedData[3] == 0x46 )
    {
     msgconfirmation = HIGH;
     flushSendData();
    }
    ParseData(); // zpracuje data fcí ParseData
   }
   else
   {
    Serial.println("ERROR: Check sum ");// odesleme text
    Serial.println("============================");// odesleme text
   }
```

```
   }
   else
   {
    //když není přijatý znak 0x03, tak připojí znak do proměnné data
    receivedData[serialCounter] = charRecived;
    serialCounter = serialCounter + 1 ;
   }
  }
}
byte ckSum(byte address, byte deviceAddress, byte var, byte units, byte dozens, byte hundreds)
{
 int data_size = 6;
 //byte data[data_size];
 byte sum = 0;

 byte ckSumData[data_size] = {
  address,
  deviceAddress,
  var,
  units,
  dozens,
  hundreds
 };
 for (int i = 0; i < data_size - 1; i++) {
  sum += ckSumData[i];
 }
 return lowByte(sum);
}

bool ckSumCheck()
{
 bool result = false;
 ckSumValue = ckSum(receivedData[1], receivedData[2], receivedData[3], receivedData[4], receivedData[5],
receivedData[6]);
 if (lowByte(receivedData[7]) - ckSumValue == 0)
 {
  result = true;
 }
 else
 {
  result = false;
 }
 return result;
}
void prepareData(byte address, byte var, byte units, byte dozens, byte hundreds, byte cksum) {
 sendData[0] = 0x02; //start byte
 sendData[1] = address; // ardesat (komu odesíláme) - u primece neni vyuzito
 sendData[2] = myAddress; //moje adresa - od koho to prijmaci doslo
 sendData[3] = var; // ocislovaní promenné
 sendData[4] = units; // hodnota jednotek
 sendData[5] = dozens; // hodnota desítek
 sendData[6] = hundreds; // hodnota stovek
 sendData[7] = lowByte(cksum); // hodnota ckSum
 sendData[8] = 0x03; // stop byte
}

void sendTelegram ()
{
 digitalWrite(SSerialTxControl, 1); // mod odesílání
 for (int i = 0; i < 9; i++) {
  RS485Serial.write(sendData[i]);
 }
 digitalWrite(SSerialTxControl, 0); // konec modu odelilaní
}
```

```
void ParseData()
{
  for (int i = 1; i < 6; i++) {
    Serial.write(receivedData[i]);
    Serial.print("/");
  }
  Serial.println("");
  Serial.println("==============================");// odesleme text
}
void flushSendData()
{
  for (int i = 0; i < 9; i++) {
    sendData[i] = "";
  }
}
void flushReceivedData()
{
  for (int i = 0; i < 9; i++) {
    receivedData[i] = "";
  }
}
```

Příloha C - Kód modulu spínacího aktoru

```
#include <SoftwareSerial.h>
#define SSerialRX        8  //Serial Receive pin
#define SSerialTX        9  //Serial Transmit pin
#define SSerialTxControl 2   // RS modul pin 3


SoftwareSerial RS485Serial(SSerialRX, SSerialTX); // RX, TX
char charRecived; // přijatá znak na RS485
byte data[9]; //zde budeme ulkádat přijatá data
int serialCounter = 0;
byte sendData[10];
//#############################################################################################################

const int pinNumber = 4;
byte myAddress = 0x32;
int LEDpin[pinNumber] = {11, 10, 6, 5};
int LEDAdress[pinNumber] = {0x31, 0x32, 0x33, 0x34};
byte pinInitialStatus[pinNumber] = {0x30, 0x30, 0x31, 0x31};
//#############################################################################################################

char device;
int deviceint;
int LEDStatus[pinNumber] = {0, 0, 0, 0};
Byte ckSumValue = 0;

void setup(){
 // Inicializujeme seriový port na klasickém pin 0 a 1 (TX a RX)
 Serial.begin(9600);
 pinMode(SSerialTxControl, OUTPUT);
 digitalWrite(SSerialTxControl, 0);  // Povolí přijímaní na modulu RS485
 for (int i = 0; i < pinNumber; i++) {
  pinMode(LEDpin[i], OUTPUT);
  Serial.print("Pin: ");
  Serial.print(LEDpin[i]);
  switchPinStatus(i, pinInitialStatus[i] );
  Serial.println(" nastaven jako output");
 }
 RS485Serial.begin(14400); // rychlost RS485 musí být nastavená stejně jako na vysílači
}

void loop()  {
 // když jsou nějaká data přijata na RS485
 if (RS485Serial.available())
 {
  charRecived = RS485Serial.read(); //Přečte přijatý znak
  if (charRecived == 0x03) // když znak je 0x03 značí to konec vyslílání
  {
   serialCounter = 0;
   if (ckSumCheck() == true)
   {
    if (data[1] == myAddress)
    {
     ckSumValue = ckSum(data[2], myAddress, 0x46, 0x30, 0x30, 0x30);
     prepareData(data[2], 0x46, 0x30, 0x30, 0x30, ckSumValue);
     sendTelegram();
     for (int i = 0; i < pinNumber; i++) {
      /*Serial.println("============================");
       Serial.write(data[3]);
       Serial.print("/");
       Serial.write(LEDAdress[i]);
       Serial.println("/");
```

```
          Serial.println("============================");*/
         if (data[3] == LEDAdress[i] )
         {
          if ( data[4] == 0x54)
          {
           togglePinStatus(i);
          }
          else if ( data[4] == 0x31 or data[4] == 0x30)
          {
           switchPinStatus(i, data[4]);
          }
          else if (data[4] == 0x48)
          {
           analogWrite(LEDpin[i], 128);
           LEDStatus[i] = 1;
           Serial.print(" LED sviti na pinu: ");// odesleme text
           Serial.print(LEDpin[i]);// odesleme text
           Serial.println();
          }
          else if (data[4] == 0x51)
          {
           analogWrite(LEDpin[i], 64);
           LEDStatus[i] = 1;
           Serial.print(" LED sviti na pinu: ");// odesleme text
           Serial.print(LEDpin[i]);// odesleme text
           Serial.println();
          }
         }
        }
       }
       ParseData(); // zpracuje data fcí ParseData
     }
     else
     {
      Serial.println("ERROR: Check sum ");// odesleme text
      Serial.println("============================");// odesleme text
     }
    }
    else
    {
     //když není přijatý znak 0x03, tak připojí znak do proměnné data
     data[serialCounter] = charRecived;
     // Serial.println(charRecived);
     serialCounter = serialCounter + 1 ;
    }
  }
}

void ParseData()
{
  for (int i = 0; i < 6; i++) {
    Serial.write(data[i]);
    Serial.print("/");
  }
  Serial.println("");
  Serial.println("============================");// odesleme text
}

void togglePinStatus(int ID)
{
  if (LEDStatus[ID] == 0)
  {
    digitalWrite(LEDpin[ID], HIGH);
    LEDStatus[ID] = 1;
    Serial.print("LED sviti na pinu: ");// odesleme text
```

```
   Serial.print(LEDpin[ID]);// odesleme text
   Serial.println();
  }
  else
  {
   digitalWrite(LEDpin[ID], LOW);
   LEDStatus[ID] = 0;
   Serial.print("LED nesviti na pinu: ");// odesleme text
   Serial.print(LEDpin[ID]);// odesleme text
   Serial.println();
  }
}

void switchPinStatus(int ID, byte status)
{
 if (status == 0x31)
 {
  digitalWrite(LEDpin[ID], HIGH);
  LEDStatus[ID] = 1;
  Serial.print("LED sviti na pinu: ");// odesleme text
  Serial.print(LEDpin[ID]);// odesleme text
  Serial.println();
 }
 else if (status == 0x30)
 {
  digitalWrite(LEDpin[ID], LOW);
  LEDStatus[ID] = 0;
  Serial.print("LED nesviti na pinu: ");// odesleme text
  Serial.print(LEDpin[ID]);// odesleme text
  Serial.println();
 }
}

byte ckSum(byte address, byte deviceAddress, byte var, byte units, byte dozens, byte hundreds)
{
 int data_size = 6;
 byte sum = 0;
 byte ckdata[data_size] = {
  address,
  deviceAddress,
  var,
  units,
  dozens,
  hundreds
 };

 for (int i = 0; i < data_size - 1; i++) {
  sum += ckdata[i];
 }
 byte crcbyte = ckdata[data_size - 1];
 return sum;
}

bool ckSumCheck()
{
 bool result = false;
 ckSumValue = ckSum(data[1], data[2], data[3], data[4], data[5], data[6]);
 if (lowByte(data[7]) - ckSumValue == 0)
 {
  result = true;
 }
 else
 {
  result = false;
 }
```

```
  return result;
}

void prepareData(byte address, byte var, byte units, byte dozens, byte hundreds, byte cksum) {
  sendData[0] = 0x02; //start byte
  sendData[1] = address; // ardesat (komu odesíláme) - u primece neni vyuzito
  sendData[2] = myAddress; //moje adresa - od koho to prijmaci doslo
  sendData[3] = var; // ocislovaní promenné
  sendData[4] = units; // hodnota jednotek
  sendData[5] = dozens; // hodnota desítek
  sendData[6] = hundreds; // hodnota stovek
  sendData[7] = lowByte(cksum); // hodnota ckSum
  sendData[8] = 0x03; // stop byte
}

void sendTelegram ()
{
  digitalWrite(SSerialTxControl, 1); // mod odesílání
  for (int i = 0; i < 9; i++) {
    RS485Serial.write(sendData[i]);
  }
  digitalWrite(SSerialTxControl, 0); // konec modu odelilaní

}
```